Package 'mlr3extralearners'

February 18, 2022

Title Extra Learners For mlr3

Version 0.5.25

Description Extra learners for use in mlr3.

License LGPL-3

Depends R (>= 3.1.0)

Imports checkmate, data.table, methods, mlr3 (>= 0.6.0), mlr3misc (>= 0.9.4), paradox, R6

Suggests actuar, apcluster, C50, coin, CoxBoost, Cubist, dbarts, distr6 (>= 1.6.0), earth, extraTrees, flexsurv, FNN, gbm, gss, keras (>= 2.3.0), kerdiest, kernlab, ks, LiblineaR, lightgbm (>= 3.0.0), locfit, logspline, mboost, mda, mgcv, mlr3cluster, mlr3learners (>= 0.4.2), mlr3proba (>= 0.4.1), mvtnorm, nnet, np, obliqueRSF, param6, partykit, penalized, pendensity, plugdensity, pracma, pseudo, randomForest, randomForestSRC, remotes, reticulate (>= 1.16), RWeka, sandwich, set6, sm, stats, survival, survivalmodels (>= 0.1.9), survivalsvm, tensorflow (>= 2.0.0), testthat, usethis

Remotes binderh/CoxBoost

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

Roxygen list(markdown = TRUE, r6 = TRUE)

RoxygenNote 7.1.2

Author Raphael Sonabend [aut] (<https://orcid.org/0000-0001-9225-4654>), Patrick Schratz [aut] (<https://orcid.org/0000-0003-0748-6624>), Lorenz A. Kapsner [ctb] (<https://orcid.org/0000-0003-1866-860X>), Lennart Schneider [ctb] (<https://orcid.org/0000-0003-4152-5308>), Stephen A Lauer [ctb] (<https://orcid.org/0000-0003-2948-630X>), Pierre Camilleri [ctb], Javier García [ctb], Sebastian Fischer [cre, aut]

Maintainer Sebastian Fischer <sebf.fischer@gmail.com>

R topics documented:

`mlr3extralearners-package` *mlr3extralearners: Extra Learners For mlr3*

Description

Extra learners for use in mlr3.

Author(s)

Maintainer: Sebastian Fischer <sebf.fischer@gmail.com>

Authors:

- Raphael Sonabend <raphaelsonabend@gmail.com> (ORCID)
- Patrick Schratz <patrick.schratz@gmail.com> (ORCID)

Other contributors:

- Lorenz A. Kapsner <lorenz.kapsner@gmail.com> (ORCID) [contributor]
- Lennart Schneider <lennart.sch@web.de> (ORCID) [contributor]
- Stephen A Lauer <stephenalauer@gmail.com> (ORCID) [contributor]
- Pierre Camilleri <camilleri_pierre@hotmail.fr> [contributor]
- Javier García <geshter@hotmail.com> [contributor]

`create_learner` *Create a New Learner*

Description

Helper function to generate all required files, and fill in fields, for new learners.

Usage

```
create_learner(
  pkg = ".",
  classname,
  algorithm,
  type,
  key = tolower(classname),
  package = tolower(classname),
  caller,
  feature_types,
  predict_types,
  properties = NULL,
  references = FALSE,
  gh_name
)
```

Arguments

pkg	character(1) Path to the mlr3extralearners package.
classname	character(1) Suffix for R6 class name passed to LearnerTypeclassname.
algorithm	character(1) Brief description of algorithm for documentation title.
type	character(1) See mlr3::mlr_reflections$task_types$type.
key	character(1) id for learner, if not provided defaults to the classname in all lower case.
package	character(1) Package from which the learner is implemented, defaults to the classname in all lower case.
caller	character(1) Training function called from the implemented package.
feature_types	character() Feature types that can be handled by the learner, see mlr3::mlr_reflections$task_feature_types.
predict_types	character() Prediction types that can be made by the learner, see mlr3::mlr_reflections$learner_predict_types
properties	character() Properties that can be handled by the learner, see mlr3::mlr_reflections$learner_properties.
references	logical(1) Set to TRUE if you want to add references for the learner.
gh_name	character(1) Your GitHub handle, used to add you as the maintainer of the learner.

Details

This function does the following:

1. Creates a learner_package_type_key.R file for the implemented learner.
2. Creates a test_package_type_key.R file for unit testing the learner.
3. Creates a test_paramtest_package_type_key.R file for testing correct implementation of learner parameters.
4. Creates a test_package.yml file for running unit tests in GitHub actions
5. Automatically completes the test (2), and yaml files (4)
6. Automatically adds the learner package to the DESCRIPTION file
7. For the learner file all fields are automatically filled but methods must be manually added along with the parameter set, this is clearly marked up in the files.

To create a learner you must follow these steps:

1. Run this function with as many arguments as possible

2. Manually add `.train`, `.predict` private methods for the learner, as well as adding the `param_set` and possibly `param_vals`. If properties include `"oob_error"` and/or `"importance"` then add these public methods manually.
3. Check the paramtests and unit tests pass locally.
4. Run
   (a) `devtools::document(roclets = c('rd','collate','namespace'))`
   (b) `styler::style_pkg(style = styler::mlr_style)` (you may need to first run remotes::)
   (c) `usethis::use_tidy_description()`
   (d) `lintr::lint_package()`
5. Open a pull request to https://github.com/mlr-org/mlr3extralearners/pulls with the new learner template.

Examples

```
## Not run:
# Simpler linear regression example
create_learner(
  classname = "LM",
  algorithm = "linear regression",
  type = "regr",
  package = "stats",
  caller = "lm",
  feature_types = c("logical", "integer", "numeric", "factor"),
  predict_types = c("response", "se"),
  properties = "weights",
  gh_name = "RaphaelS1"
)

# Slightly more complex random forest learner
create_learner(
  classname = "RandomForestSRC",
  algorithm = "random forest",
  type = "surv",
  package = "randomForestSRC",
  caller = "rfsrc",
  feature_types = c("logical", "integer", "numeric", "factor"),
  predict_types = c("crank", "distr"),
  properties = c("importance", "missings", "oob_error", "weights"),
  references = TRUE,
  gh_name = "RaphaelS1"
)

## End(Not run)
```

`install_catboost` *Install catboost*

Description

Helper function to install catboost

Usage

```
install_catboost(
  version = NULL,
  os = NULL,
  install_required = TRUE,
  INSTALL_opts = c("--no-multiarch", "--no-test-load"),
  ...
)
```

Arguments

version	(character(1)) Version to install, if NULL installs latest
os	(character(1)) Operating system to install on, if NULL automatically detected
install_required	(logical(1)) If TRUE (default) then installs required packages: curl, jsonlite, remotes
INSTALL_opts	(character()) Passed to remotes::install_url
...	ANY Other arguments passed to remotes::install_url

---

install_learners *Install Learner Dependencies*

Description

Install required dependencies for specified learners. Works for packages on GitHub and CRAN, otherwise must be manually installed.

Usage

```
install_learners(.keys, repos = "https://cloud.r-project.org", ...)
```

Arguments

.keys	(character()) Keys passed to mlr_learners specifying learners to install.
repos	(character(1)) Passed to utils::install.packages.
...	(ANY) Additional options to pass to utils::install.packages or remotes::install_github.

list_mlr3learners *List Learners in mlr3verse*

Description

Lists all learners, properties, and associated packages in a table that can be filtered and queried.

Usage

```
list_mlr3learners(select = NULL, filter = NULL)
```

Arguments

select	character() Passed to data.table::subset.
filter	list() Named list of conditions to filter on, names correspond to column names in table.

Examples

```
list_mlr3learners(
  select = c("id", "properties", "predict_types"),
  filter = list(class = "surv", predict_types = "distr"))
```

lrn *Syntactic Sugar for Learner Construction*

Description

Overloads mlr3::lrn to automatically detect if required packages are installed.

Usage

```
lrn(.key, ...)

lrns(.keys, ...)
```

Arguments

.key	(character(1)) Key passed to mlr_learners to retrieve the learner.
...	ANY Passed to mlr3::lrns
.keys	(character()) Keys passed to mlr_learners to retrieve the learners.

`mlr_learners_classif.AdaBoostM1`
Classification AdaBoostM1 Learner

Description

Calls RWeka::AdaBoostM1 from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.AdaBoostM1")
lrn("classif.AdaBoostM1")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
P	integer	100		$[90, 100]$
Q	logical	FALSE	TRUE, FALSE	-
S	integer	1		$[1, \infty)$
I	integer	10		$[1, \infty)$
W	untyped	DecisionStump		-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifAdaBoostM1`

Methods

Public methods:

- `LearnerClassifAdaBoostM1$new()`
- `LearnerClassifAdaBoostM1$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifAdaBoostM1$new()
```

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

Usage:

```
LearnerClassifAdaBoostM1$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Freund, Yoav, Schapire, E R, others (1996). "Experiments with a new boosting algorithm." In *icml*, volume 96, 148–156. Citeseer.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.AdaBoostM1")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.bart

Classification BART (Bayesian Additive Regression Trees) Learner

---

Description

Calls dbarts::bart from package **dbarts**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.bart")
lrn("classif.bart")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **dbarts**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	200		$[1, \infty)$
k	numeric	2		$[0, \infty)$
power	numeric	2		$[0, \infty)$
base	numeric	0.95		$[0, 1]$
binaryOffset	numeric	0		$(-\infty, \infty)$
ndpost	integer	1000		$[1, \infty)$
nskip	integer	100		$[0, \infty)$
printevery	integer	100		$[0, \infty)$
keepevery	integer	1		$[1, \infty)$
keeptrainfits	logical	TRUE	TRUE, FALSE	-
usequants	logical	FALSE	TRUE, FALSE	-
numcut	integer	100		$[1, \infty)$
printcutoffs	integer	0		$(-\infty, \infty)$
verbose	logical	TRUE	TRUE, FALSE	-
keepcall	logical	TRUE	TRUE, FALSE	-
sampleronly	logical	FALSE	TRUE, FALSE	-
seed	integer	NA		$(-\infty, \infty)$
proposalprobs	untyped			-

Custom mlr3 defaults

- Parameter: keeptrees
- Original: FALSE
- New: TRUE
- Reason: Required for prediction
- Parameter: offset
- The parameter is removed, because only `dbarts::bart2` allows an offset during training, and therefore the offset parameter in `dbarts:::predict.bart` is irrelevant for `dbarts::dbart`.
- Parameter: nthread, nchain, combineChains, combinechains
- The parameters are removed as parallelization of multiple models is handled by future.
- Parameter: sigest, sigdf, sigquant, keeptres
- Regression only.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifBart`

Methods

Public methods:

- `LearnerClassifBart$new()`
- `LearnerClassifBart$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifBart$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

ck37

References

Sparapani, Rodney, Spanbauer, Charles, McCulloch, Robert (2021). "Nonparametric machine learning and efficient computation with bayesian additive regression trees: the BART R package." *Journal of Statistical Software*, **97**, 1–66.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("dbarts", quietly = TRUE)) {
  learner = mlr3::lrn("classif.bart")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.C50`
Classification C5.0 Learner

Description

Calls C50::C5.0 from package **C50**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.C50")
lrn("classif.C50")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **C50**

Parameters

Id	Type	Default	Levels	Range
trials	integer	1		$[1, \infty)$
rules	logical	FALSE	TRUE, FALSE	-
costs	untyped			-
subset	logical	TRUE	TRUE, FALSE	-
bands	integer	-		$[0, 1000]$
winnow	logical	FALSE	TRUE, FALSE	-
noGlobalPruning	logical	FALSE	TRUE, FALSE	-
CF	numeric	0.25		$[0, 1]$
minCases	integer	2		$[0, \infty)$
fuzzyThreshold	logical	FALSE	TRUE, FALSE	-
sample	numeric	0		$[0, 0.999]$
seed	integer	-		$(-\infty, \infty)$
earlyStopping	logical	TRUE	TRUE, FALSE	-
label	untyped	outcome		-
na.action	untyped	:: , stats , na.pass		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifC50`

Methods

Public methods:

- `LearnerClassifC50$new()`
- `LearnerClassifC50$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifC50$new()
```

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

Usage:

```
LearnerClassifC50$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Quinlan, Ross J (2014). *C4. 5: programs for machine learning*. Elsevier.

`mlr_learners_classif.catboost`

Gradient Boosted Decision Trees Classification Learner

Description

Calls catboost::catboost.train from package **catboost**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.catboost")
lrn("classif.catboost")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”, “prob”
- Feature Types: “numeric”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3extralearners**, **catboost**

Parameters

Id	Type	Default	Levels
loss_function_twoclass	character	Logloss	Logloss, CrossEntropy
loss_function_multiclass	character	MultiClass	MultiClass, MultiClassOneVsAll
iterations	integer	1000	
learning_rate	numeric	0.03	
random_seed	integer	0	
l2_leaf_reg	numeric	3	
bootstrap_type	character	-	Bayesian, Bernoulli, MVS, Poisson, No
bagging_temperature	numeric	1	
subsample	numeric	-	
sampling_frequency	character	PerTreeLevel	PerTree, PerTreeLevel
sampling_unit	character	Object	Object, Group
mvs_reg	numeric	-	
random_strength	numeric	1	
depth	integer	6	
grow_policy	character	SymmetricTree	SymmetricTree, Depthwise, Lossguide
min_data_in_leaf	integer	1	
max_leaves	integer	31	
ignored_features	untyped		
one_hot_max_size	untyped	FALSE	
has_time	logical	FALSE	TRUE, FALSE
rsm	numeric	1	
nan_mode	character	Min	Min, Max
fold_permutation_block	integer	-	
leaf_estimation_method	character	-	Newton, Gradient, Exact
leaf_estimation_iterations	integer	-	
leaf_estimation_backtracking	character	AnyImprovement	No, AnyImprovement, Armijo
fold_len_multiplier	numeric	2	
approx_on_full_history	logical	TRUE	TRUE, FALSE
class_weights	untyped	-	
auto_class_weights	character	None	None, Balanced, SqrtBalanced
boosting_type	character	-	Ordered, Plain
boost_from_average	logical	-	TRUE, FALSE
langevin	logical	FALSE	TRUE, FALSE
diffusion_temperature	numeric	10000	
score_function	character	Cosine	Cosine, L2, NewtonCosine, NewtonL2
monotone_constraints	untyped	-	
feature_weights	untyped	-	
first_feature_use_penalties	untyped	-	

penalties_coefficient	numeric	1	
per_object_feature_penalties	untyped	-	
model_shrink_rate	numeric	-	
model_shrink_mode	character	-	Constant, Decreasing
target_border	numeric	-	
border_count	integer	-	
feature_border_type	character	GreedyLogSum	Median, Uniform, UniformAndQuantiles, MaxLogSum, Mi
per_float_feature_quantization	untyped	-	
classes_count	integer	-	
thread_count	integer	1	
task_type	character	CPU	CPU, GPU
devices	untyped	-	
logging_level	character	Silent	Silent, Verbose, Info, Debug
metric_period	integer	1	
train_dir	untyped	catboost_info	
model_size_reg	numeric	0.5	
allow_writing_files	logical	FALSE	TRUE, FALSE
save_snapshot	logical	FALSE	TRUE, FALSE
snapshot_file	untyped	-	
snapshot_interval	integer	600	
simple_ctr	untyped	-	
combinations_ctr	untyped	-	
ctr_target_border_count	integer	-	
counter_calc_method	character	Full	SkipTest, Full
max_ctr_complexity	integer	-	
ctr_leaf_count_limit	integer	-	
store_all_simple_ctr	logical	FALSE	TRUE, FALSE
final_ctr_computation_mode	character	Default	Default, Skip
verbose	logical	FALSE	TRUE, FALSE
ntree_start	integer	0	
ntree_end	integer	0	

Installation

The easiest way to install catboost is with the helper function install_catboost.

Custom mlr3 defaults

- `logging_level`:
  - Actual default: "Verbose"
  - Adjusted default: "Silent"
  - Reason for change: consistent with other mlr3 learners
- `thread_count`:
  - Actual default: -1

  - Adjusted default: 1
  - Reason for change: consistent with other mlr3 learners
- `allow_writing_files`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: consistent with other mlr3 learners
- `save_snapshot`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: consistent with other mlr3 learners

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifCatboost`

Methods

Public methods:

- `LearnerClassifCatboost$new()`
- `LearnerClassifCatboost$importance()`
- `LearnerClassifCatboost$clone()`

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

Usage:

```
LearnerClassifCatboost$new()
```

Method `importance()`**:** The importance scores are calculated using `catboost.get_feature_importance`, setting `type = "FeatureImportance"`, returned for 'all'.

Usage:

```
LearnerClassifCatboost$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Dorogush, Veronika A, Ershov, Vasily, Gulin, Andrey (2018). "CatBoost: gradient boosting with categorical features support." *arXiv preprint arXiv:1810.11363*.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("catboost", quietly = TRUE)) {
  learner = mlr3::lrn("classif.catboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.cforest

Classification Conditional Random Forest Learner

Description

Calls partykit::cforest from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.cforest")
lrn("classif.cforest")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
replace	logical	FALSE	TRUE, FALSE	-
fraction	numeric	0.632		$[0, 1]$
mtry	integer	-		$[0, \infty)$
mtryratio	numeric	-		$[0, 1]$
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
trace	logical	FALSE	TRUE, FALSE	-
offset	untyped	-		-
cluster	untyped	-		-
scores	untyped	-		-
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Univariate	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
nmax	untyped	-		-
pargs	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0		$[0, 1]$
logmincriterion	numeric	0		$(-\infty, \infty)$
minsplit	integer	20		$[1, \infty)$
minbucket	integer	7		$[1, \infty)$
minprob	numeric	0.01		$[0, 1]$
stump	logical	FALSE	TRUE, FALSE	-
lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
nresample	integer	9999		$[1, \infty)$
tol	numeric	1.490116e-08		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$
numsurrogate	logical	FALSE	TRUE, FALSE	-
maxdepth	integer	Inf		$[0, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[0, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	TRUE	TRUE, FALSE	-
saveinfo	logical	FALSE	TRUE, FALSE	-
update	logical	FALSE	TRUE, FALSE	-
splitflavour	character	ctree	ctree, exhaustive	-
maxvar	integer	-		$[1, \infty)$
OOB	logical	FALSE	TRUE, FALSE	-
simplify	logical	TRUE	TRUE, FALSE	-
scale	logical	TRUE	TRUE, FALSE	-
nperm	integer	1		$[0, \infty)$
risk	character	loglik	loglik, misclassification	-

conditional	logical	FALSE	TRUE, FALSE	-
threshold	numeric	0.2		$(-\infty, \infty)$

Custom mlr3 defaults

- `mtry`:
  - This hyperparameter can alternatively be set via the added hyperparameter `mtryratio` as `mtry = max(ceiling(mtryratio * n_features), 1)`. Note that `mtry` and `mtryratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifCForest`

Methods

Public methods:

- `LearnerClassifCForest$new()`
- `LearnerClassifCForest$oob_error()`
- `LearnerClassifCForest$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifCForest$new()
```

Method `oob_error()`**:** The importance scores are calculated using `partykit::varimp()`. The out-of-bag error, calculated using the OOB predictions from `partykit`.

Usage:

```
LearnerClassifCForest$oob_error()
```

Returns: `numeric(1)`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("partykit", quietly = TRUE) && requireNamespace("sandwich", quietly = TRUE) && requireNamespa
  learner = mlr3::lrn("classif.cforest")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.ctree`

Classification Conditional Inference Tree Learner

Description

Calls partykit::ctree from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.ctree")
lrn("classif.ctree")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”, “prob”
- Feature Types: “integer”, “numeric”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Bonferroni	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
nmax	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0.95		$[0, 1]$
logmincriterion	numeric	-		$(-\infty, \infty)$
minsplit	integer	20		$[1, \infty)$
minbucket	integer	7		$[1, \infty)$
minprob	numeric	0.01		$[0, 1]$
stump	logical	FALSE	TRUE, FALSE	-
lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
nresample	integer	9999		$[1, \infty)$
tol	numeric	-		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$
numsurrogate	logical	FALSE	TRUE, FALSE	-
mtry	integer	Inf		$[0, \infty)$
maxdepth	integer	Inf		$[0, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[0, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	FALSE	TRUE, FALSE	-
maxvar	integer	-		$[1, \infty)$
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
saveinfo	logical	TRUE	TRUE, FALSE	-
update	logical	FALSE	TRUE, FALSE	-
splitflavour	character	ctree	ctree, exhaustive	-
offset	untyped	-		-
cluster	untyped	-		-
scores	untyped	-		-
doFit	logical	TRUE	TRUE, FALSE	-
maxpts	integer	25000		$(-\infty, \infty)$
abseps	numeric	0.001		$[0, \infty)$
releps	numeric	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifCTree`

Methods

Public methods:

- `LearnerClassifCTree$new()`
- `LearnerClassifCTree$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifCTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("partykit", quietly = TRUE) && requireNamespace("sandwich", quietly = TRUE) && requireNamespa
  learner = mlr3::lrn("classif.ctree")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.earth`

Classification MARS (Multivariate Adaptive Regression Splines) Learner

---

Description

Calls earth::earth from package **earth**.

Details

Methods for variance estimations are not yet implemented.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.earth")
lrn("classif.earth")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **earth**

Parameters

Id	Type	Default	Levels	Range
wp	untyped			-
offset	untyped			-
keepxy	logical	FALSE	TRUE, FALSE	-
trace	character	0	0, .3, .5, 1, 2, 3, 4, 5	-
degree	integer	1		$[1, \infty)$
penalty	numeric	2		$[-1, \infty)$

nk	untyped			-
thresh	numeric	0.001		$(-\infty, \infty)$
minspan	numeric	0		$[0, \infty)$
endspan	numeric	0		$[0, \infty)$
newvar.penalty	numeric	0		$[0, \infty)$
fast.k	integer	20		$[0, \infty)$
fast.beta	integer	1		$[0, 1]$
linpreds	untyped	FALSE		-
allowed	untyped	-		-
pmethod	character	backward	backward, none, exhaustive, forward, seqrep, cv	-
nprune	integer	-		$[0, \infty)$
nfold	integer	0		$[0, \infty)$
ncross	integer	1		$[0, \infty)$
stratify	logical	TRUE	TRUE, FALSE	-
varmod.method	character	none	none, const, lm, rlm, earth, gam, power, power0, x.lm, x.rlm, ...	-
varmod.exponent	numeric	1		$(-\infty, \infty)$
varmod.conv	numeric	1		$[0, 1]$
varmod.clamp	numeric	0.1		$(-\infty, \infty)$
varmod.minspan	numeric	-3		$(-\infty, \infty)$
Scale.y	logical	FALSE	TRUE, FALSE	-
Adjust.endspan	numeric	2		$(-\infty, \infty)$
Auto.linpreds	logical	TRUE	TRUE, FALSE	-
Force.weights	logical	FALSE	TRUE, FALSE	-
Use.beta.cache	logical	TRUE	TRUE, FALSE	-
Force.xtx.prune	logical	FALSE	TRUE, FALSE	-
Get.leverages	logical	TRUE	TRUE, FALSE	-
Exhaustive.tol	numeric	1e-10		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifEarth`

Methods

Public methods:

- `LearnerClassifEarth$new()`
- `LearnerClassifEarth$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifEarth$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

pkopper

References

Milborrow, Stephen, Hastie, T, Tibshirani, R (2014). "Earth: multivariate adaptive regression spline models." *R package version*, **3**(7).

Friedman, H J (1991). "Multivariate adaptive regression splines." *The annals of statistics*, **19**(1), 1–67.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("earth", quietly = TRUE)) {
  learner = mlr3::lrn("classif.earth")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.extratrees

Classification ExtraTrees Learner

Description

Calls extraTrees::extraTrees from package **extraTrees**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.extratrees")
lrn("classif.extratrees")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **extraTrees**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
nodesize	integer	1		$[1, \infty)$
numRandomCuts	integer	1		$(-\infty, \infty)$
evenCuts	logical	FALSE	TRUE, FALSE	-
numThreads	integer	1		$[1, \infty)$
subsetSizes	untyped	-		-
subsetGroups	untyped	-		-
tasks	untyped	-		-
probOfTaskCuts	numeric	-		$[0, 1]$
numRandomTaskCuts	integer	1		$[1, \infty)$
na.action	character	stop	stop, zero, fuse	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifExtraTrees`

Methods

Public methods:

- `LearnerClassifExtraTrees$new()`
- `LearnerClassifExtraTrees$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifExtraTrees$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Geurts, Pierre, Ernst, Damien, Wehenkel, Louis (2006). "Extremely randomized trees." *Machine learning*, **63**(1), 3–42.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("extraTrees", quietly = TRUE)) {
  learner = mlr3::lrn("classif.extratrees")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.fnn`

Classification Fast Nearest Neighbor Search Learner

Description

Calls FNN::knn from package **FNN**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.fnn")
lrn("classif.fnn")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **FNN**

Parameters

Id	Type	Default	Levels	Range
k	integer	1		$[1, \infty)$
algorithm	character	kd_tree	kd_tree, cover_tree, brute	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifFNN`

Methods

Public methods:

- `LearnerClassifFNN$new()`
- `LearnerClassifFNN$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifFNN$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Boltz, Sylvain, Debreuve, Eric, Barlaud, Michel (2007). "kNN-based high-dimensional Kullback-Leibler distance for tracking." In *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07)*, 16–16. IEEE.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("FNN", quietly = TRUE)) {
  learner = mlr3::lrn("classif.fnn")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.gam

Classification Generalized Additive Model Learner

Description

Generalized additive models. Calls mgcv::gam from package **mgcv**.

Multiclass classification is not implemented yet.

A gam formula specific to the task at hand is required for the `formula` parameter (see example and `?mgcv::formula.gam`). Beware, if no formula is provided, a fallback formula is used that will make the gam behave like a glm (this behavior is required for the unit tests). Only features specified in the formula will be used, superseding columns with col_roles "feature" in the task.

Calls mgcv::gam from package **mgcv**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.gam")
lrn("classif.gam")
```

Meta Information

- Task type: "classif"
- Predict Types: "prob", "response"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **mgcv**

Parameters

Id	Type	Default	Levels	Range
formula	untyped	-		-
offset	untyped			-
method	character	GCV.Cp	GCV.Cp, GACV.Cp, REML, P-REML, ML, P-ML	-
optimizer	untyped	c , outer , newton		-
scale	numeric	0		$(-\infty, \infty)$
select	logical	FALSE	TRUE, FALSE	-
knots	untyped			-
sp	untyped			-
min.sp	untyped			-
H	untyped			-
gamma	numeric	1		$[1, \infty)$
paraPen	untyped			-
G	untyped			-
in.out	untyped			-
drop.unused.levels	logical	TRUE	TRUE, FALSE	-
drop.intercept	logical	FALSE	TRUE, FALSE	-
nthreads	integer	1		$[1, \infty)$
irls.reg	numeric	0		$[0, \infty)$
epsilon	numeric	1e-07		$[0, \infty)$
maxit	integer	200		$(-\infty, \infty)$
trace	logical	FALSE	TRUE, FALSE	-
mgcv.tol	numeric	1e-07		$[0, \infty)$
mgcv.half	integer	15		$[0, \infty)$
rank.tol	numeric	1.490116e-08		$[0, \infty)$
nlm	untyped	list		-
optim	untyped	list		-
newton	untyped	list		-
outerPIsteps	integer	0		$[0, \infty)$
idLinksBases	logical	TRUE	TRUE, FALSE	-
scalePenalty	logical	TRUE	TRUE, FALSE	-

efs.lspmax	integer	15		$[0, \infty)$
efs.tol	numeric	0.1		$[0, \infty)$
scale.est	character	fletcher	fletcher, pearson, deviance	-
edge.correct	logical	FALSE	TRUE, FALSE	-
block.size	integer	1000		$(-\infty, \infty)$
unconditional	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifGam`

Methods

Public methods:

- `LearnerClassifGam$new()`
- `LearnerClassifGam$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifGam$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

JazzyPierrot

References

Hastie, J T, Tibshirani, J R (2017). *Generalized additive models*. Routledge.

Wood, Simon (2012). "mgcv: Mixed GAM Computation Vehicle with GCV/AIC/REML smoothness estimation."

Examples

```
# simple example
t = mlr3::tsk("spam")
l = mlr3::lrn("classif.gam")
l$param_set$values$formula = type ~ s(george) + s(charDollar) + s(edu) + ti(george, edu)
l$train(t)
l$model
```

`mlr_learners_classif.gamboost` *Boosted Generalized Additive Classification Learner*

Description

Calls mboost::gamboost from package **mboost**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.gamboost")
lrn("classif.gamboost")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **mboost**

Parameters

Id	Type	Default	Levels	Range
baselearner	character	bbs	bbs, bols, btree	-
dfbase	integer	4		$(-\infty, \infty)$
offset	numeric	NULL		$(-\infty, \infty)$
family	character	Binomial	Binomial, AdaExp, AUC, custom	-
custom.family	untyped	-		-
link	character	logit	logit, probit	-
type	character	adaboost	glm, adaboost	-
mstop	integer	100		$(-\infty, \infty)$
nu	numeric	0.1		$(-\infty, \infty)$
risk	character	inbag	inbag, oobag, none	-
oobweights	untyped			-
trace	logical	FALSE	TRUE, FALSE	-
stopintern	untyped	FALSE		-
na.action	untyped	:: , stats , na.omit		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifGAMBoost`

Methods

Public methods:

- `LearnerClassifGAMBoost$new()`
- `LearnerClassifGAMBoost$clone()`

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

Usage:

```
LearnerClassifGAMBoost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mboost", quietly = TRUE)) {
  learner = mlr3::lrn("classif.gamboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.gausspr`

Classifession Gaussian Process Learner

Description

Calls kernlab::gausspr from package **kernlab**.

Details

Parameters `sigma`, `degree`, `scale`, `offset` and `order` are added to make tuning `kpar` easier. If `kpar` is provided then these new parameters are ignored. If none are provided then the default "automatic" is used for `kpar`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.gausspr")
lrn("classif.gausspr")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”, “prob”
- Feature Types: “numeric”, “integer”, “logical”, “character”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3extralearners**, **kernlab**

Parameters

Id	Type	Default	Levels	Range
scaled	untyped	TRUE		-
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot	-
sigma	numeric	-		$(-\infty, \infty$
degree	numeric	-		$(-\infty, \infty$
scale	numeric	-		$(-\infty, \infty$

offset	numeric	-		$(-\infty, \infty$
order	numeric	-		$(-\infty, \infty$
kpar	untyped	automatic		-
tol	numeric	0.001		$[0, \infty)$
fit	logical	TRUE	TRUE, FALSE	-
na.action	untyped	na.omit		-
coupler	character	minpair	minpair, pkpd	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifGausspr`

Methods

Public methods:

- `LearnerClassifGausspr$new()`
- `LearnerClassifGausspr$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifGausspr$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**(9), 1–20.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.

- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("kernlab", quietly = TRUE)) {
  learner = mlr3::lrn("classif.gausspr")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.gbm`

Classification Gradient Boosting Machine Learner

Description

Calls gbm::gbm from package **gbm**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.gbm")
lrn("classif.gbm")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **gbm**

Parameters

Id	Type	Default	Levels	Range
distribution	character	bernoulli	bernoulli, adaboost, huberized, multinomial	-
n.trees	integer	100		$[1, \infty)$
interaction.depth	integer	1		$[1, \infty)$
n.minobsinnode	integer	10		$[1, \infty)$
shrinkage	numeric	0.001		$[0, \infty)$

bag.fraction	numeric	0.5		$[0, 1]$
train.fraction	numeric	1		$[0, 1]$
cv.folds	integer	0		$(-\infty, \infty)$
keep.data	logical	FALSE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-
n.cores	integer	1		$(-\infty, \infty)$
var.monotone	untyped	-		-

Custom mlr3 defaults

- `keep_data`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: `keep_data` = `FALSE` saves memory during model fitting.
- `n.cores`:
  - Actual default: NULL
  - Adjusted default: 1
  - Reason for change: Suppressing the automatic internal parallelization if `cv.folds` > 0.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifGBM`

Methods

Public methods:

- `LearnerClassifGBM$new()`
- `LearnerClassifGBM$importance()`
- `LearnerClassifGBM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifGBM$new()
```

Method `importance()`**:** The importance scores are extracted by `gbm::relative.influence()` from the model.

Usage:

```
LearnerClassifGBM$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Friedman, H J (2002). "Stochastic gradient boosting." *Computational statistics & data analysis*, **38**(4), 367–378.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("gbm", quietly = TRUE)) {
  learner = mlr3::lrn("classif.gbm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.glmboost

Boosted Generalized Linear Classification Learner

Description

Calls mboost::glmboost from package **mboost**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.glmboost")
lrn("classif.glmboost")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **mboost**

Parameters

Id	Type	Default	Levels	Range
offset	numeric	NULL		$(-\infty, \infty)$
family	character	Binomial	Binomial, AdaExp, AUC, custom	-
custom.family	untyped	-		-
link	character	logit	logit, probit	-
type	character	adaboost	glm, adaboost	-
center	logical	TRUE	TRUE, FALSE	-
mstop	integer	100		$(-\infty, \infty)$
nu	numeric	0.1		$(-\infty, \infty)$
risk	character	inbag	inbag, oobag, none	-
oobweights	untyped			-
trace	logical	FALSE	TRUE, FALSE	-
stopintern	untyped	FALSE		-
na.action	untyped	:: , stats , na.omit		-
contrasts.arg	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifGLMBoost`

Methods

Public methods:

- `LearnerClassifGLMBoost$new()`
- `LearnerClassifGLMBoost$clone()`

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

Usage:

```
LearnerClassifGLMBoost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mboost", quietly = TRUE)) {
  learner = mlr3::lrn("classif.glmboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.IBk`

Classification IBk Learner

Description

Calls RWeka::IBk from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.IBk")
lrn("classif.IBk")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”, “prob”
- Feature Types: “numeric”, “factor”, “ordered”, “integer”
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
I	logical	FALSE	TRUE, FALSE	-
F	logical	FALSE	TRUE, FALSE	-
K	integer	1		$[1, \infty)$
E	logical	FALSE	TRUE, FALSE	-
W	integer	0		$[0, \infty)$
X	logical	FALSE	TRUE, FALSE	-
A	untyped	weka.core.neighboursearch.LinearNNSearch		-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifIBk`

Methods

Public methods:

- `LearnerClassifIBk$new()`
- `LearnerClassifIBk$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifIBk$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Aha, W D, Kibler, Dennis, Albert, K M (1991). "Instance-based learning algorithms." *Machine learning*, **6**(1), 37–66.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.IBk")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.J48`
Classification J48 Learner

Description

Calls RWeka::J48 from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.J48")
lrn("classif.J48")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
U	logical	FALSE	TRUE, FALSE	-
O	logical	FALSE	TRUE, FALSE	-
C	numeric	0.25		$[2.22044604925031e-16, 1]$
M	integer	2		$[1, \infty)$
R	logical	FALSE	TRUE, FALSE	-
N	integer	3		$[2, \infty)$
B	logical	FALSE	TRUE, FALSE	-
S	logical	FALSE	TRUE, FALSE	-
L	logical	FALSE	TRUE, FALSE	-
A	logical	FALSE	TRUE, FALSE	-
J	logical	FALSE	TRUE, FALSE	-
Q	integer	1		$[1, \infty)$
doNotMakeSplitPointActualValue	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$

options	untyped	-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifJ48`

Methods

Public methods:

- `LearnerClassifJ48$new()`
- `LearnerClassifJ48$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifJ48$new()
```

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

Usage:

```
LearnerClassifJ48$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Quinlan, Ross J (2014). *C4. 5: programs for machine learning*. Elsevier.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.J48")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.JRip

Classification JRip Learner

Description

Calls RWeka::JRip from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.JRip")
lrn("classif.JRip")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
F	integer	3		$[2, \infty)$
N	numeric	2		$[0, \infty)$
O	integer	2		$[1, \infty)$
D	logical	FALSE	TRUE, FALSE	-
S	integer	1		$[1, \infty)$
E	logical	FALSE	TRUE, FALSE	-
P	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifJRip`

Methods

Public methods:

- `LearnerClassifJRip$new()`
- `LearnerClassifJRip$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifJRip$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Cohen, W W (1995). "Fast effective rule induction." In *Machine learning proceedings 1995*, 115–123. Elsevier.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.JRip")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.ksvm`

Classification Kernlab Support Vector Machine

Description

Calls kernlab::ksvm from package **kernlab**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.ksvm")
lrn("classif.ksvm")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **kernlab**

Parameters

Id	Type	Default	Levels	Range
scaled	logical	TRUE	TRUE, FALSE	-
type	character	C-svc	C-svc, nu-svc, C-bsvc, spoc-svc, kbb-svc	-
kernel	character	rbfdot	rbfdot, polydot, vanilladot, laplacedot, besseldot, anovadot	-
C	numeric	1		$(-\infty, \infty)$
nu	numeric	0.2		$[0, \infty)$
cache	integer	40		$[1, \infty)$
tol	numeric	0.001		$[0, \infty)$
shrinking	logical	TRUE	TRUE, FALSE	-
sigma	numeric	-		$[0, \infty)$
degree	integer	-		$[1, \infty)$
scale	numeric	-		$[0, \infty)$
order	integer	-		$(-\infty, \infty)$
offset	numeric	-		$(-\infty, \infty)$
coupler	character	minpair	minpair, pkpd	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifKSVM`

Methods

Public methods:

- `LearnerClassifKSVM$new()`
- `LearnerClassifKSVM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifKSVM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

mboecker

References

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**(9), 1–20.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("kernlab", quietly = TRUE)) {
  learner = mlr3::lrn("classif.ksvm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.liblinear`
LiblineaR Classification Learner

Description

Calls LiblineaR::LiblineaR from package **LiblineaR**.

Details

Type of SVC depends on `type` argument:

- `0` – L2-regularized logistic regression (primal)
- `1` - L2-regularized L2-loss support vector classification (dual)
- `3` - L2-regularized L1-loss support vector classification (dual)
- `2` – L2-regularized L2-loss support vector classification (primal)
- `4` – Support vector classification by Crammer and Singer
- `5` - L1-regularized L2-loss support vector classification
- `6` - L1-regularized logistic regression
- `7` - L2-regularized logistic regression (dual)

If number of records > number of features, `type = 2` is faster than `type = 1` (Hsu et al. 2003).

Note that probabilistic predictions are only available for types `0`, `6`, and `7`. The default `epsilon` value depends on the `type` parameter, see LiblineaR::LiblineaR.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.liblinear")
lrn("classif.liblinear")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **LiblineaR**

Parameters

Id	Type	Default	Levels	Range
type	integer	0		$[0, 7]$
cost	numeric	1		$[0, \infty)$
epsilon	numeric	-		$[0, \infty)$
bias	numeric	1		$(-\infty, \infty)$
cross	integer	0		$[0, \infty)$
verbose	logical	FALSE	TRUE, FALSE	-
wi	untyped			-
findC	logical	FALSE	TRUE, FALSE	-
useInitC	logical	TRUE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifLiblineaR`

Methods

Public methods:

- `LearnerClassifLiblineaR$new()`
- `LearnerClassifLiblineaR$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifLiblineaR$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, Lin, Chih-Jen (2008). "LIBLINEAR: A library for large linear classification." *the Journal of machine Learning research*, **9**, 1871–1874.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("LiblineaR", quietly = TRUE)) {
  learner = mlr3::lrn("classif.liblinear")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.lightgbm

Classification Light GBM Learner

Description

Calls lightgbm::lgb.train from package **lightgbm**.

Details

For categorical features either pre-process data by encoding columns or specify the categorical columns with the `categorical_feature` parameter. For this learner please do not prefix the categorical feature with name:. Instead of providing the data that is used for early stopping explicitly, the parameter `early_stopping_split` determines the proportion of the training data that is used for early stopping.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.lightgbm")
lrn("classif.lightgbm")
```

Meta Information

- Task type: "classif"
- Predict Types: "prob", "response"
- Feature Types: "numeric", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **lightgbm**

Parameters

Id	Type	Default	Levels
nrounds	integer	5	
objective	character	binary	binary, multiclass, multiclassova, cross_entropy
metric	character		, None, ndcg, map, auc, average_precision, bin
custom_eval	untyped		
verbose	integer	1	
record	logical	TRUE	TRUE, FALSE
eval_freq	integer	1	
init_model	untyped		
early_stopping_rounds	integer	-	
early_stopping_split	numeric	0	
callbacks	untyped	-	
reset_data	logical	FALSE	TRUE, FALSE
categorical_feature	untyped		
boosting	character	gbdt	gbdt, rf, dart, goss
linear_tree	logical	FALSE	TRUE, FALSE
num_iterations	integer	100	
learning_rate	numeric	0.1	
num_leaves	integer	31	
tree_learner	character	serial	serial, feature, data, voting
num_threads	integer	0	
device_type	character	cpu	cpu, gpu
seed	integer	-	
deterministic	logical	FALSE	TRUE, FALSE
force_col_wise	logical	FALSE	TRUE, FALSE
force_row_wise	logical	FALSE	TRUE, FALSE
histogram_pool_size	integer	-1	
max_depth	integer	-1	
min_data_in_leaf	integer	20	
min_sum_hessian_in_leaf	numeric	0.001	
bagging_fraction	numeric	1	

pos_bagging_fraction	numeric	1	
neg_bagging_fraction	numeric	1	
bagging_freq	integer	0	
bagging_seed	integer	3	
feature_fraction	numeric	1	
feature_fraction_bynode	numeric	1	
feature_fraction_seed	integer	2	
extra_trees	logical	FALSE	TRUE, FALSE
extra_seed	integer	6	
first_metric_only	logical	FALSE	TRUE, FALSE
max_delta_step	numeric	0	
lambda_l1	numeric	0	
lambda_l2	numeric	0	
linear_lambda	numeric	0	
min_gain_to_split	numeric	0	
drop_rate	numeric	0.1	
max_drop	integer	50	
skip_drop	numeric	0.5	
xgboost_dart_mode	logical	FALSE	TRUE, FALSE
uniform_drop	logical	FALSE	TRUE, FALSE
drop_seed	integer	4	
top_rate	numeric	0.2	
other_rate	numeric	0.1	
min_data_per_group	integer	100	
max_cat_threshold	integer	32	
cat_l2	numeric	10	
cat_smooth	numeric	10	
max_cat_to_onehot	integer	4	
top_k	integer	20	
monotone_constraints	untyped		
monotone_constraints_method	character	basic	basic, intermediate, advanced
monotone_penalty	numeric	0	
feature_contri	untyped		
forcedsplits_filename	untyped		
refit_decay_rate	numeric	0.9	
cegb_tradeoff	numeric	1	
cegb_penalty_split	numeric	0	
cegb_penalty_feature_lazy	untyped	-	
cegb_penalty_feature_coupled	untyped	-	
path_smooth	numeric	0	
interaction_constraints	untyped	-	
input_model	untyped		
output_model	untyped	LightGBM_model.txt	
saved_feature_importance_type	integer	0	
snapshot_freq	integer	-1	
max_bin	integer	255	
max_bin_by_feature	untyped		
min_data_in_bin	integer	3	

bin_construct_sample_cnt	integer	200000	
data_random_seed	integer	1	
is_enable_sparse	logical	TRUE	TRUE, FALSE
enable_bundle	logical	TRUE	TRUE, FALSE
use_missing	logical	TRUE	TRUE, FALSE
zero_as_missing	logical	FALSE	TRUE, FALSE
feature_pre_filter	logical	TRUE	TRUE, FALSE
pre_partition	logical	FALSE	TRUE, FALSE
two_round	logical	FALSE	TRUE, FALSE
header	logical	FALSE	TRUE, FALSE
group_column	untyped		
forcedbins_filename	untyped		
save_binary	logical	FALSE	TRUE, FALSE
objective_seed	integer	5	
is_unbalance	logical	FALSE	TRUE, FALSE
scale_pos_weight	numeric	1	
sigmoid	numeric	1	
boost_from_average	logical	TRUE	TRUE, FALSE
lambdarank_truncation_level	integer	30	
lambdarank_norm	logical	TRUE	TRUE, FALSE
label_gain	untyped	-	
metric_freq	integer	1	
is_provide_training_metric	logical	FALSE	TRUE, FALSE
eval_at	untyped	:, 1, 5	
multi_error_top_k	integer	1	
auc_mu_weights	untyped		
num_machines	integer	1	
local_listen_port	integer	12400	
time_out	integer	120	
machine_list_filename	untyped		
machines	untyped		
gpu_platform_id	integer	-1	
gpu_device_id	integer	-1	
gpu_use_dp	logical	FALSE	TRUE, FALSE
num_gpu	integer	1	
start_iteration	integer	0	
num_iteration	integer	-1	
pred_early_stop	logical	FALSE	TRUE, FALSE
pred_early_stop_freq	integer	10	
pred_early_stop_margin	numeric	10	
output_result	untyped	LightGBM_predict_result.txt	

Custom mlr3 defaults

- `num_threads`:

  - Actual default: 0L
  - Adjusted default: 1L
  - Reason for change: Prevents accidental conflicts with `future`.
- `verbose`:
  - Actual default: 1L
  - Adjusted default: -1L
  - Reason for change: Prevents accidental conflicts with mlr messaging system.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifLightGBM`

Methods

Public methods:

- `LearnerClassifLightGBM$new()`
- `LearnerClassifLightGBM$importance()`
- `LearnerClassifLightGBM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifLightGBM$new()
```

Method `importance()`**:** The importance scores are extracted from `lbg.importance`.

Usage:

```
LearnerClassifLightGBM$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

kapsner

References

Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei, Liu, Tie-Yan (2017). "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems*, **30**.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("lightgbm", quietly = TRUE)) {
  learner = mlr3::lrn("classif.lightgbm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_classif.LMT

Classification Logistic Model Trees Learner

Description

Calls RWeka::LMT from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.LMT")
lrn("classif.LMT")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
B	logical	FALSE	TRUE, FALSE	-
R	logical	FALSE	TRUE, FALSE	-
C	logical	FALSE	TRUE, FALSE	-
P	logical	FALSE	TRUE, FALSE	-
I	integer	-		$[1, \infty)$
M	integer	15		$[1, \infty)$
W	numeric	0		$[0, 1]$
A	logical	FALSE	TRUE, FALSE	-
doNotMakeSplitPointActualValue	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifLMT`

Methods

Public methods:

- `LearnerClassifLMT$new()`
- `LearnerClassifLMT$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifLMT$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Landwehr, Niels, Hall, Mark, Frank, Eibe (2005). "Logistic model trees." *Machine learning*, **59**(1), 161–205.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.LMT")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.lssvm` *Classification Least Squares Support Vector Machine Learner*

Description

Calls kernlab::lssvm from package **kernlab**.

Details

Parameters `sigma`, `degree`, `scale`, `offset`, `order`, `length`, `lambda`, and `normalized` are added to make tuning `kpar` easier. If `kpar` is provided then these new parameters are ignored. If none are provided then the default "automatic" is used for `kpar`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.lssvm")
lrn("classif.lssvm")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”
- Feature Types: “numeric”, “integer”
- Required Packages: **mlr3**, **mlr3extralearners**, **kernlab**

Parameters

Id	Type	Default	Levels
scaled	untyped	TRUE	
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot, stringdo
sigma	numeric	-	
degree	numeric	-	
scale	numeric	-	
offset	numeric	-	
order	numeric	-	
length	integer	-	
lambda	numeric	-	
normalized	logical	-	TRUE, FALSE
kpar	untyped	automatic	
tau	numeric	0.01	
reduced	logical	TRUE	TRUE, FALSE
rank	integer	-	

delta	integer	40	
tol	numeric	1e-04	
fit	logical	TRUE	TRUE, FALSE
na.action	untyped	na.omit	
coupler	character	minpair	minpair, pkpd

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifLSSVM`

Methods

Public methods:

- `LearnerClassifLSSVM$new()`
- `LearnerClassifLSSVM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifLSSVM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**(9), 1–20.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("kernlab", quietly = TRUE)) {
  learner = mlr3::lrn("classif.lssvm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.mob`

Classification Model-based Recursive Partitioning Learner

Description

Calls partykit::mob from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.mob")
lrn("classif.mob")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
rhs	untyped	-		-
fit	untyped	-		-
offset	untyped	-		-
cluster	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
bonferroni	logical	TRUE	TRUE, FALSE	-
minsize	integer	-		$[1, \infty)$
minsplit	integer	-		$[1, \infty)$
minbucket	integer	-		$[1, \infty)$
maxdepth	integer	Inf		$[0, \infty)$
mtry	integer	Inf		$[0, \infty)$

trim	numeric	0.1		$[0, \infty)$
breakties	logical	FALSE	TRUE, FALSE	-
parm	untyped	-		-
dfsplit	integer	-		$[0, \infty)$
prune	untyped	-		-
restart	logical	TRUE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-
caseweights	logical	TRUE	TRUE, FALSE	-
ytype	character	vector	vector, matrix, data.frame	-
xtype	character	matrix	vector, matrix, data.frame	-
terminal	untyped	object		-
inner	untyped	object		-
model	logical	TRUE	TRUE, FALSE	-
numsplit	character	left	left, center	-
catsplit	character	binary	binary, multiway	-
vcov	character	opg	opg, info, sandwich	-
ordinal	character	chisq	chisq, max, L2	-
nrep	integer	10000		$[0, \infty)$
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
additional	untyped	-		-
predict_fun	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifMob`

Methods

Public methods:

- `LearnerClassifMob$new()`
- `LearnerClassifMob$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifMob$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("partykit", quietly = TRUE) && requireNamespace("sandwich", quietly = TRUE) && requireNamespa
  learner = mlr3::lrn("classif.mob")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.OneR`

Classification OneR Learner

Description

Calls RWeka::OneR from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.OneR")
lrn("classif.OneR")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
B	integer	6		$[1, \infty)$
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifOneR`

Methods

Public methods:

- `LearnerClassifOneR$new()`
- `LearnerClassifOneR$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifOneR$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Holte, C R (1993). "Very simple classification rules perform well on most commonly used datasets." *Machine learning*, **11**(1), 63–90.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.OneR")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.PART`
Classification PART Learner

Description

Calls RWeka::PART from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.PART")
lrn("classif.PART")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
C	numeric	0.25		$[2.22044604925031e-16, 1]$
M	integer	2		$[1, \infty)$
R	logical	FALSE	TRUE, FALSE	-
N	integer	3		$[1, \infty)$
B	logical	FALSE	TRUE, FALSE	-
U	logical	FALSE	TRUE, FALSE	-
J	logical	FALSE	TRUE, FALSE	-
Q	integer	1		$[1, \infty)$
doNotMakeSplitPointActualValue	logical	FALSE	TRUE, FALSE	-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifPART`

Methods

Public methods:

- `LearnerClassifPART$new()`
- `LearnerClassifPART$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifPART$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Frank, Eibe, Witten, H I (1998). "Generating accurate rule sets without global optimization."

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("classif.PART")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.randomForest`

Classification Random Forest Learner

Description

Calls randomForest::randomForest from package **randomForest**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.randomForest")
lrn("classif.randomForest")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **randomForest**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
replace	logical	TRUE	TRUE, FALSE	-
classwt	untyped			-
cutoff	untyped	-		-
strata	untyped	-		-
sampsize	untyped	-		-
nodesize	integer	1		$[1, \infty)$
maxnodes	integer	-		$[1, \infty)$
importance	character	FALSE	accuracy, gini, none, FALSE	-
localImp	logical	FALSE	TRUE, FALSE	-
proximity	logical	FALSE	TRUE, FALSE	-
oob.prox	logical	-	TRUE, FALSE	-
norm.votes	logical	TRUE	TRUE, FALSE	-
do.trace	logical	FALSE	TRUE, FALSE	-
keep.forest	logical	TRUE	TRUE, FALSE	-
keep.inbag	logical	FALSE	TRUE, FALSE	-
predict.all	logical	FALSE	TRUE, FALSE	-
nodes	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifRandomForest`

Methods

Public methods:

- `LearnerClassifRandomForest$new()`
- `LearnerClassifRandomForest$importance()`
- `LearnerClassifRandomForest$oob_error()`
- `LearnerClassifRandomForest$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifRandomForest$new()
```

Method `importance()`**:** The importance scores are extracted from the slot `importance`. Parameter 'importance' must be set to either `"accuracy"` or `"gini"`.

Usage:

```
LearnerClassifRandomForest$importance()
```

Returns: Named `numeric()`.

Method `oob_error()`**:** OOB errors are extracted from the model slot `err.rate`.

Usage:

```
LearnerClassifRandomForest$oob_error()
```

Returns: `numeric(1)`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

pat-s

References

Breiman, Leo (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("randomForest", quietly = TRUE)) {
  learner = mlr3::lrn("classif.randomForest")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_classif.rfsrc`

Classification Random Forest SRC Learner

Description

Calls randomForestSRC::rfsrc from package **randomForestSRC**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("classif.rfsrc")
lrn("classif.rfsrc")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "logical", "integer", "numeric", "factor"
- Required Packages: **mlr3**, **mlr3extralearners**, **randomForestSRC**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	1000		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
mtry.ratio	numeric	-		$[0, 1]$
nodesize	integer	15		$[1, \infty)$
nodedepth	integer	-		$[1, \infty)$
splitrule	character	gini	gini, auc, entropy	-
nsplit	integer	10		$[0, \infty)$
importance	character	FALSE	FALSE, TRUE, none, permute, random, anti	-
block.size	integer	10		$[1, \infty)$
bootstrap	character	by.root	by.root, by.node, none, by.user	-
samptype	character	swor	swor, swr	-
samp	untyped	-		-
membership	logical	FALSE	TRUE, FALSE	-
sampsize	untyped	-		-
sampsize.ratio	numeric	-		$[0, 1]$
na.action	character	na.omit	na.omit, na.impute	-
nimpute	integer	1		$[1, \infty)$
ntime	integer	-		$[1, \infty)$
cause	integer	-		$[1, \infty)$

proximity	character	FALSE	FALSE, TRUE, inbag, oob, all	-
distance	character	FALSE	FALSE, TRUE, inbag, oob, all	-
forest.wt	character	FALSE	FALSE, TRUE, inbag, oob, all	-
xvar.wt	untyped	-		-
split.wt	untyped	-		-
forest	logical	TRUE	TRUE, FALSE	-
var.used	character	FALSE	FALSE, all.trees, by.tree	-
split.depth	character	FALSE	FALSE, all.trees, by.tree	-
seed	integer	-		$(-\infty, -1]$
do.trace	logical	FALSE	TRUE, FALSE	-
statistics	logical	FALSE	TRUE, FALSE	-
get.tree	untyped	-		-
outcome	character	train	train, test	-
ptn.count	integer	0		$[0, \infty)$
cores	integer	1		$[1, \infty)$

Custom mlr3 defaults

- `cores`:
  - Actual default: Auto-detecting the number of cores
  - Adjusted default: 1
  - Reason for change: Threading conflicts with explicit parallelization via **future**.
- `mtry`:
  - This hyperparameter can alternatively be set via the added hyperparameter `mtry.ratio` as `mtry = max(ceiling(mtry.ratio * n_features), 1)`. Note that `mtry` and `mtry.ratio` are mutually exclusive.
- `sampsize`:
  - This hyperparameter can alternatively be set via the added hyperparameter `sampsize.ratio` as `sampsize = max(ceiling(sampsize.ratio * n_obs), 1)`. Note that `sampsize` and `sampsize.ratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifRandomForestSRC`

Methods

Public methods:

- `LearnerClassifRandomForestSRC$new()`
- `LearnerClassifRandomForestSRC$importance()`
- `LearnerClassifRandomForestSRC$selected_features()`
- `LearnerClassifRandomForestSRC$oob_error()`
- `LearnerClassifRandomForestSRC$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerClassifRandomForestSRC$new()
```

Method `importance()`**:** The importance scores are extracted from the model slot `importance`, returned for 'all'.

Usage:

```
LearnerClassifRandomForestSRC$importance()
```

Returns: Named `numeric()`.

Method `selected_features()`**:** Selected features are extracted from the model slot `var.used`.

Usage:

```
LearnerClassifRandomForestSRC$selected_features()
```

Returns: `character()`.

Method `oob_error()`**:** OOB error extracted from the model slot `err.rate`.

Usage:

```
LearnerClassifRandomForestSRC$oob_error()
```

Returns: `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Breiman, Leo (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("randomForestSRC", quietly = TRUE)) {
  learner = mlr3::lrn("classif.rfsrc")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.kde_kd`

Density Kerdiest Kernel Learner

Description

Calls kerdiest::kde from package **kerdiest**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.kde_kd")
lrn("dens.kde_kd")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **kerdiest**

Parameters

Id	Type	Default	Levels	Range
bw	numeric	-		$[0, \infty)$
type_kernel	character	n	n, e, t, b	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensKDEkd`

Methods

Public methods:

- `LearnerDensKDEkd$new()`
- `LearnerDensKDEkd$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensKDEkd$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Quintela-del-Río, Alejandro, Estévez-Pérez G (2012). "Nonparametric kernel distribution function estimation with kerdiest: an R package for bandwidth choice and applications." *Journal of Statistical Software*, **50**, 1–21.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("kerdiest", quietly = TRUE)) {
  learner = mlr3::lrn("dens.kde_kd")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.kde_ks`
Density KS Kernel Learner

Description

Calls ks::kde from package **ks**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.kde_ks")
lrn("dens.kde_ks")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **ks**

Parameters

Id	Type	Default	Levels	Range
h	numeric	-		$[0, \infty)$
H	untyped	-		-
gridsize	untyped	-		-
gridtype	untyped	-		-
xmin	numeric	-		$(-\infty, \infty)$
xmax	numeric	-		$(-\infty, \infty)$
supp	numeric	3.7		$(-\infty, \infty)$
binned	numeric	-		$(-\infty, \infty)$
bgridsize	untyped	-		-
positive	logical	FALSE	TRUE, FALSE	-
adj.positive	untyped	-		-
w	untyped	-		-
compute.cont	logical	TRUE	TRUE, FALSE	-
approx.cont	logical	TRUE	TRUE, FALSE	-
unit.interval	logical	FALSE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensKDEks`

Methods

Public methods:

- `LearnerDensKDEks$new()`
- `LearnerDensKDEks$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensKDEks$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Gramacki, Artur, Gramacki, Jarosław (2017). "FFT-based fast computation of multivariate kernel density estimators with unconstrained bandwidth matrices." *Journal of Computational and Graphical Statistics*, **26**(2), 459–462.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("ks", quietly = TRUE)) {
  learner = mlr3::lrn("dens.kde_ks")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.locfit`

Density Locfit Learner

Description

Calls locfit::density.lf from package **locfit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.locfit")
lrn("dens.locfit")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **locfit**

Parameters

Id	Type	Default	Levels	Range
window	character	gaus	tcub, rect, trwt, tria, epan, bisq, gaus	-
width	numeric	-		$(-\infty, \infty)$
from	numeric	-		$(-\infty, \infty)$
to	numeric	-		$(-\infty, \infty)$
cut	numeric	-		$(-\infty, \infty)$
deg	numeric	0		$(-\infty, \infty)$
link	character	ident	ident, log, logit, inverse, sqrt, arcsin	-
kern	character	tcub	rect, trwt, tria, epan, bisq, gauss, tcub	-
kt	character	sph	sph, prod	-
renorm	logical	FALSE	TRUE, FALSE	-
maxk	integer	100		$[0, \infty)$

itype	character	-	prod, mult, mlin, haz	-
mint	integer	20		$[1, \infty)$
maxit	integer	20		$[1, \infty)$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensLocfit`

Methods

Public methods:

- `LearnerDensLocfit$new()`
- `LearnerDensLocfit$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensLocfit$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Loader, Clive (2006). *Local regression and likelihood*. Springer Science & Business Media.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("locfit", quietly = TRUE)) {
  learner = mlr3::lrn("dens.locfit")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.logspline`
Density Logspline Learner

Description

Calls logspline::logspline from package **logspline**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.logspline")
lrn("dens.logspline")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf", "cdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **logspline**

Parameters

Id	Type	Default	Levels	Range
lbound	numeric	-		$(-\infty, \infty)$
ubound	numeric	-		$(-\infty, \infty)$
maxknots	numeric	0		$[0, \infty)$
knots	untyped	-		-
nknots	numeric	0		$[0, \infty)$
penalty	untyped	-		-
silent	logical	TRUE	TRUE, FALSE	-
mind	numeric	-1		$(-\infty, \infty)$
error.action	integer	2		$[0, 2]$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensLogspline`

Methods

Public methods:

- `LearnerDensLogspline$new()`
- `LearnerDensLogspline$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensLogspline$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Kooperberg, Charles, Stone, J C (1992). "Logspline density estimation for censored data." *Journal of Computational and Graphical Statistics*, **1**(4), 301–328.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("logspline", quietly = TRUE)) {
  learner = mlr3::lrn("dens.logspline")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.mixed`

Density Mixed Data Kernel Learner

Description

Calls np::npudens from package **np**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.mixed")
lrn("dens.mixed")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **np**

Parameters

Id	Type	Default	Levels	Range
bws	untyped	-		-
ckertype	character	gaussian	gaussian, epanechnikov, uniform	-
bwscaling	logical	FALSE	TRUE, FALSE	-
bwmethod	character	cv.ml	cv.ml, cv.ls, normal-reference	-
bwtype	character	fixed	fixed, generalized_nn, adaptive_nn	-
bandwidth.compute	logical	FALSE	TRUE, FALSE	-
ckerorder	integer	2		$[2, 8]$
remin	logical	TRUE	TRUE, FALSE	-
itmax	integer	10000		$[1, \infty)$
nmulti	integer	-		$[1, \infty)$
ftol	numeric	1.490116e-07		$(-\infty, \infty)$

tol	numeric	0.0001490116		$(-\infty, \infty)$
small	numeric	1.490116e-05		$(-\infty, \infty)$
lbc.dir	numeric	0.5		$(-\infty, \infty)$
dfc.dir	numeric	0.5		$(-\infty, \infty)$
cfac.dir	untyped	* , 2.5 , (3 - sqrt(5))		-
initc.dir	numeric	1		$(-\infty, \infty)$
lbd.dir	numeric	0.1		$(-\infty, \infty)$
hbd.dir	numeric	1		$(-\infty, \infty)$
dfac.dir	untyped	* , 0.25 , (3 - sqrt(5))		-
initd.dir	numeric	1		$(-\infty, \infty)$
lbc.init	numeric	0.1		$(-\infty, \infty)$
hbc.init	numeric	2		$(-\infty, \infty)$
cfac.init	numeric	0.5		$(-\infty, \infty)$
lbd.init	numeric	0.1		$(-\infty, \infty)$
hbd.init	numeric	0.9		$(-\infty, \infty)$
dfac.init	numeric	0.37		$(-\infty, \infty)$
ukertype	character	-	aitchisonaitken, liracine	-
okertype	character	-	wangvanryzin, liracine	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensMixed`

Methods

Public methods:

- `LearnerDensMixed$new()`
- `LearnerDensMixed$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensMixed$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Li, Qi, Racine, Jeff (2003). "Nonparametric estimation of distributions with categorical and continuous data." *journal of multivariate analysis*, **86**(2), 266–292.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("np", quietly = TRUE)) {
  learner = mlr3::lrn("dens.mixed")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.nonpar`

Density Nonparametric Learner

Description

Calls sm::sm.density from package **sm**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.nonpar")
lrn("dens.nonpar")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **sm**

Parameters

Id	Type	Default	Levels	Range
h	numeric	-		$(-\infty, \infty)$
group	untyped	-		-
delta	numeric	-		$(-\infty, \infty)$
h.weights	numeric	1		$(-\infty, \infty)$
hmult	untyped	1		-
method	character	normal	normal, cv, sj, df, aicc	-
positive	logical	FALSE	TRUE, FALSE	-
verbose	untyped	1		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensNonparametric`

Methods

Public methods:

- `LearnerDensNonparametric$new()`
- `LearnerDensNonparametric$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensNonparametric$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Bowman, A.W., Azzalini, A. (1997). *Applied Smoothing Techniques for Data Analysis: The Kernel Approach with S-Plus Illustrations*, series Oxford Statistical Science Series. OUP Oxford. ISBN 9780191545696, https://books.google.de/books?id=7WBMrZ9umRYC.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("sm", quietly = TRUE)) {
  learner = mlr3::lrn("dens.nonpar")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_dens.pen *Density Penalized Learner*

Description

Calls pendensity::pendensity from package **pendensity**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.pen")
lrn("dens.pen")
```

Meta Information

- Task type: “dens”
- Predict Types: “pdf”, “cdf”
- Feature Types: “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **pendensity**

Parameters

Id	Type	Default	Levels	Range
base	character	bspline	bspline, gaussian	-
no.base	numeric	41		$(-\infty, \infty)$
max.iter	numeric	20		$(-\infty, \infty)$
lambda0	numeric	500		$(-\infty, \infty)$
q	numeric	3		$(-\infty, \infty)$
sort	logical	TRUE	TRUE, FALSE	-
with.border	untyped	-		-
m	numeric	3		$(-\infty, \infty)$
eps	numeric	0.01		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensPenalized`

Methods

Public methods:

- `LearnerDensPenalized$new()`
- `LearnerDensPenalized$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensPenalized$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Schellhase, Christian, Kauermann, Göran (2012). "Density estimation and comparison with a penalized mixture approach." *Computational Statistics*, **27**(4), 757–777.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("pendensity", quietly = TRUE)) {
  learner = mlr3::lrn("dens.pen")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_dens.plug

Density Plug-In Kernel Learner

Description

Calls plugdensity::plugin.density from package **plugdensity**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.plug")
lrn("dens.plug")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf"
- Feature Types: "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **plugdensity**

Parameters

Id	Type	Default	Levels
na.rm	logical	FALSE	TRUE, FALSE

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensPlugin`

Methods

Public methods:

- `LearnerDensPlugin$new()`
- `LearnerDensPlugin$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensPlugin$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Engel, Joachim, Herrmann, Eva, Gasser, Theo (1994). "An iterative bandwidth selector for kernel estimation of densities and their derivatives." *Journaltitle of Nonparametric Statistics*, **4**(1), 21–34.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("plugdensity", quietly = TRUE)) {
  learner = mlr3::lrn("dens.plug")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_dens.spline`

Density Smoothing Splines Learner

Description

Calls gss::ssden from package **gss**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("dens.spline")
lrn("dens.spline")
```

Meta Information

- Task type: "dens"
- Predict Types: "pdf", "cdf"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **gss**

Parameters

Id	Type	Default	Levels	Range
type	untyped	-		-
alpha	numeric	1.4		$(-\infty, \infty)$
weights	untyped	-		-
na.action	untyped	:: , stats , na.omit		-
id.basis	untyped	-		-
nbasis	integer	-		$(-\infty, \infty)$
seed	numeric	-		$(-\infty, \infty)$
domain	untyped	-		-
quad	untyped	-		-
qdsz.depth	numeric	-		$(-\infty, \infty)$
bias	untyped	-		-
prec	numeric	1e-07		$(-\infty, \infty)$
maxiter	integer	30		$[1, \infty)$
skip.iter	logical	-	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerDens` -> `LearnerDensSpline`

Methods

Public methods:

- `LearnerDensSpline$new()`
- `LearnerDensSpline$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerDensSpline$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Gu, Chong, Wang, Jingyuan (2003). "Penalized likelihood density estimation: Direct cross-validation and scalable approximation." *Statistica Sinica*, 811–826.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("gss", quietly = TRUE)) {
  learner = mlr3::lrn("dens.spline")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.bart`

Regression BART (Bayesian Additive Regression Trees) Learner

Description

Calls dbarts::bart from package **dbarts**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.bart")
lrn("regr.bart")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **dbarts**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	200		$[1, \infty)$
sigest	untyped			-
sigdf	integer	3		$[1, \infty)$
sigquant	numeric	0.9		$[0, 1]$
k	numeric	2		$[0, \infty)$
power	numeric	2		$[0, \infty)$
base	numeric	0.95		$[0, 1]$
ndpost	integer	1000		$[1, \infty)$
nskip	integer	100		$[0, \infty)$
printevery	integer	100		$[0, \infty)$
keepevery	integer	1		$[1, \infty)$
keeptrainfits	logical	TRUE	TRUE, FALSE	-
usequants	logical	FALSE	TRUE, FALSE	-
numcut	integer	100		$[1, \infty)$
printcutoffs	integer	0		$(-\infty, \infty)$
verbose	logical	TRUE	TRUE, FALSE	-
keeptrees	logical	FALSE	TRUE, FALSE	-
keepcall	logical	TRUE	TRUE, FALSE	-
sampleronly	logical	FALSE	TRUE, FALSE	-
seed	integer	NA		$(-\infty, \infty)$
proposalprobs	untyped			-

Custom mlr3 defaults

- Parameter: keeptrees
  - Original: FALSE
  - New: TRUE
  - Reason: Required for prediction
- Parameter: offset
  - The parameter is removed, because only `dbarts::bart2` allows an offset during training, and therefore the offset parameter in `dbarts:::predict.bart` is irrelevant for `dbarts::dbart`.
- Parameter: nthread, nchain, combineChains, combinechains
  - The parameters are removed as parallelization of multiple models is handled by future.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrBart`

Methods

Public methods:

- `LearnerRegrBart$new()`
- `LearnerRegrBart$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrBart$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

ck37

References

Sparapani, Rodney, Spanbauer, Charles, McCulloch, Robert (2021). "Nonparametric machine learning and efficient computation with bayesian additive regression trees: the BART R package." *Journal of Statistical Software*, **97**, 1–66.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("dbarts", quietly = TRUE)) {
  learner = mlr3::lrn("regr.bart")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.catboost`

Gradient Boosted Decision Trees Regression Learner

Description

Calls catboost::catboost.train from package **catboost**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.catboost")
lrn("regr.catboost")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **catboost**

Parameters

Id	Type	Default	Levels
loss_function	character	RMSE	MAE, MAPE, Poisson, Quantile, RMSE, LogLinQuantile, I
iterations	integer	1000	
learning_rate	numeric	0.03	
random_seed	integer	0	
l2_leaf_reg	numeric	3	
bootstrap_type	character	-	Bayesian, Bernoulli, MVS, Poisson, No
bagging_temperature	numeric	1	
subsample	numeric	-	
sampling_frequency	character	PerTreeLevel	PerTree, PerTreeLevel
sampling_unit	character	Object	Object, Group
mvs_reg	numeric	-	
random_strength	numeric	1	
depth	integer	6	
grow_policy	character	SymmetricTree	SymmetricTree, Depthwise, Lossguide
min_data_in_leaf	integer	1	
max_leaves	integer	31	
has_time	logical	FALSE	TRUE, FALSE
rsm	numeric	1	
nan_mode	character	Min	Min, Max

fold_permutation_block	integer	-	
leaf_estimation_method	character	-	Newton, Gradient, Exact
leaf_estimation_iterations	integer	-	
leaf_estimation_backtracking	character	AnyImprovement	No, AnyImprovement, Armijo
fold_len_multiplier	numeric	2	
approx_on_full_history	logical	TRUE	TRUE, FALSE
boosting_type	character	-	Ordered, Plain
boost_from_average	logical	-	TRUE, FALSE
langevin	logical	FALSE	TRUE, FALSE
diffusion_temperature	numeric	10000	
score_function	character	Cosine	Cosine, L2, NewtonCosine, NewtonL2
monotone_constraints	untyped	-	
feature_weights	untyped	-	
first_feature_use_penalties	untyped	-	
penalties_coefficient	numeric	1	
per_object_feature_penalties	untyped	-	
model_shrink_rate	numeric	-	
model_shrink_mode	character	-	Constant, Decreasing
target_border	numeric	-	
border_count	integer	-	
feature_border_type	character	GreedyLogSum	Median, Uniform, UniformAndQuantiles, MaxLogSum, Mi
per_float_feature_quantization	untyped	-	
thread_count	integer	1	
task_type	character	CPU	CPU, GPU
devices	untyped	-	
logging_level	character	Silent	Silent, Verbose, Info, Debug
metric_period	integer	1	
train_dir	untyped	catboost_info	
model_size_reg	numeric	0.5	
allow_writing_files	logical	FALSE	TRUE, FALSE
save_snapshot	logical	FALSE	TRUE, FALSE
snapshot_file	untyped	-	
snapshot_interval	integer	600	
simple_ctr	untyped	-	
combinations_ctr	untyped	-	
ctr_target_border_count	integer	-	
counter_calc_method	character	Full	SkipTest, Full
max_ctr_complexity	integer	-	
ctr_leaf_count_limit	integer	-	
store_all_simple_ctr	logical	FALSE	TRUE, FALSE
final_ctr_computation_mode	character	Default	Default, Skip
verbose	logical	FALSE	TRUE, FALSE
ntree_start	integer	0	
ntree_end	integer	0	

Installation

The easiest way to install catboost is with the helper function install_catboost.

Custom mlr3 defaults

- `logging_level`:
  - Actual default: "Verbose"
  - Adjusted default: "Silent"
  - Reason for change: consistent with other mlr3 learners
- `thread_count`:
  - Actual default: -1
  - Adjusted default: 1
  - Reason for change: consistent with other mlr3 learners
- `allow_writing_files`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: consistent with other mlr3 learners
- `save_snapshot`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: consistent with other mlr3 learners

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrCatboost`

Methods

Public methods:

- `LearnerRegrCatboost$new()`
- `LearnerRegrCatboost$importance()`
- `LearnerRegrCatboost$clone()`

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

Usage:

```
LearnerRegrCatboost$new()
```

Method `importance()`**:** The importance scores are calculated using `catboost.get_feature_importance`, setting `type = "FeatureImportance"`, returned for 'all'.

Usage:

```
LearnerRegrCatboost$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Dorogush, Veronika A, Ershov, Vasily, Gulin, Andrey (2018). "CatBoost: gradient boosting with categorical features support." *arXiv preprint arXiv:1810.11363*.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("catboost", quietly = TRUE)) {
  learner = mlr3::lrn("regr.catboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_regr.cforest

Regression Conditional Random Forest Learner

Description

Calls partykit::cforest from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.cforest")
lrn("regr.cforest")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
replace	logical	FALSE	TRUE, FALSE	-
fraction	numeric	0.632		$[0, 1]$
mtry	integer	-		$[0, \infty)$
mtryratio	numeric	-		$[0, 1]$
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
trace	logical	FALSE	TRUE, FALSE	-
offset	untyped	-		-
cluster	untyped	-		-
scores	untyped	-		-
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Univariate	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
nmax	untyped	-		-
pargs	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0		$[0, 1]$
logmincriterion	numeric	0		$(-\infty, \infty)$
minsplit	integer	20		$[1, \infty)$
minbucket	integer	7		$[1, \infty)$
minprob	numeric	0.01		$[0, 1]$
stump	logical	FALSE	TRUE, FALSE	-
lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
maxvar	integer	-		$[1, \infty)$
nresample	integer	9999		$[1, \infty)$
tol	numeric	1.490116e-08		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$

numsurrogate	logical	FALSE	TRUE, FALSE	-
maxdepth	integer	Inf		$[0, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[0, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	TRUE	TRUE, FALSE	-
saveinfo	logical	FALSE	TRUE, FALSE	-
update	logical	FALSE	TRUE, FALSE	-
splitflavour	character	ctree	ctree, exhaustive	-
OOB	logical	FALSE	TRUE, FALSE	-
simplify	logical	TRUE	TRUE, FALSE	-
scale	logical	TRUE	TRUE, FALSE	-
nperm	integer	1		$[0, \infty)$
risk	character	loglik	loglik, misclassification	-
conditional	logical	FALSE	TRUE, FALSE	-
threshold	numeric	0.2		$(-\infty, \infty)$

Custom mlr3 defaults

- `mtry`:
  - This hyperparameter can alternatively be set via the added hyperparameter `mtryratio` as `mtry = max(ceiling(mtryratio * n_features), 1)`. Note that `mtry` and `mtryratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrCForest`

Methods

Public methods:

- `LearnerRegrCForest$new()`
- `LearnerRegrCForest$oob_error()`
- `LearnerRegrCForest$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrCForest$new()
```

Method `oob_error()`**:** The out-of-bag error, calculated using the OOB predictions from `partykit`.

Usage:

```
LearnerRegrCForest$oob_error()
```

Returns: `numeric(1)`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("partykit", quietly = TRUE) && requireNamespace("sandwich", quietly = TRUE) && requireNamespa
  learner = mlr3::lrn("regr.cforest")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.ctree`

Regression Conditional Inference Tree Learner

Description

Calls partykit::ctree from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.ctree")
lrn("regr.ctree")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Bonferroni	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
nmax	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0.95		$[0, 1]$
logmincriterion	numeric	-		$(-\infty, \infty)$
minsplit	integer	20		$[1, \infty)$
minbucket	integer	7		$[1, \infty)$
minprob	numeric	0.01		$[0, \infty)$
stump	logical	FALSE	TRUE, FALSE	-
lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
maxvar	integer	-		$[1, \infty)$
nresample	integer	9999		$[1, \infty)$
tol	numeric	-		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$
numsurrogate	logical	FALSE	TRUE, FALSE	-

mtry	integer	Inf		$[0, \infty)$
maxdepth	integer	Inf		$[0, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[0, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	FALSE	TRUE, FALSE	-
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
saveinfo	logical	TRUE	TRUE, FALSE	-
update	logical	FALSE	TRUE, FALSE	-
splitflavour	character	ctree	ctree, exhaustive	-
offset	untyped	-		-
cluster	untyped	-		-
scores	untyped	-		-
doFit	logical	TRUE	TRUE, FALSE	-
maxpts	integer	25000		$(-\infty, \infty)$
abseps	numeric	0.001		$[0, \infty)$
releps	numeric	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrCTree`

Methods

Public methods:

- `LearnerRegrCTree$new()`
- `LearnerRegrCTree$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrCTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("partykit", quietly = TRUE) && requireNamespace("sandwich", quietly = TRUE) && requireNamespa
  learner = mlr3::lrn("regr.ctree")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.cubist`

Regression Cubist Learner

Description

Calls Cubist::cubist from package **Cubist**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.cubist")
lrn("regr.cubist")
```

Meta Information

- Task type: “regr”
- Predict Types: “response”
- Feature Types: “integer”, “numeric”, “character”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3extralearners**, **Cubist**

Parameters

Id	Type	Default	Levels	Range
committees	integer	1		$[1, 100]$
unbiased	logical	FALSE	TRUE, FALSE	-
rules	integer	100		$[1, \infty)$
extrapolation	numeric	100		$[0, 100]$
sample	integer	0		$[0, \infty)$
seed	integer	794		$(-\infty, \infty)$
label	untyped	outcome		-
neighbors	integer	0		$[0, 9]$

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrCubist`

Methods

Public methods:

- `LearnerRegrCubist$new()`
- `LearnerRegrCubist$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrCubist$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Quinlan, R J, others (1992). "Learning with continuous classes." In *5th Australian joint conference on artificial intelligence*, volume 92, 343–348. World Scientific.

Quinlan, Ross J (1993). "Combining instance-based and model-based learning." In *Proceedings of the tenth international conference on machine learning*, 236–243.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("Cubist", quietly = TRUE)) {
  learner = mlr3::lrn("regr.cubist")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.earth`

Regression MARS (Multivariate Adaptive Regression Splines) Learner

---

Description

Calls earth::earth from package **earth**.

Details

Methods for variance estimations are not yet implemented.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.earth")
lrn("regr.earth")
```

Meta Information

- Task type: “regr”
- Predict Types: “response”, “se”
- Feature Types: “numeric”, “factor”, “integer”
- Required Packages: **mlr3**, **mlr3extralearners**, **earth**

Parameters

Id	Type	Default	Levels	Range
wp	untyped			-
offset	untyped			-
keepxy	logical	FALSE	TRUE, FALSE	-
trace	character	0	0, .3, .5, 1, 2, 3, 4, 5	-
degree	integer	1		$[1, \infty)$
penalty	numeric	2		$[-1, \infty)$
nk	untyped			-
thresh	numeric	0.001		$(-\infty, \infty)$
minspan	numeric	0		$[0, \infty)$
endspan	numeric	0		$[0, \infty)$
newvar.penalty	numeric	0		$[0, \infty)$
fast.k	integer	20		$[0, \infty)$
fast.beta	integer	1		$[0, 1]$
linpreds	untyped	FALSE		-
allowed	untyped	-		-
pmethod	character	backward	backward, none, exhaustive, forward, seqrep, cv	-
nprune	integer	-		$[0, \infty)$
nfold	integer	0		$[0, \infty)$
ncross	integer	1		$[0, \infty)$
stratify	logical	TRUE	TRUE, FALSE	-
varmod.method	character	none	none, const, lm, rlm, earth, gam, power, power0, x.lm, x.rlm, ...	-
varmod.exponent	numeric	1		$(-\infty, \infty)$
varmod.conv	numeric	1		$[0, 1]$
varmod.clamp	numeric	0.1		$(-\infty, \infty)$
varmod.minspan	numeric	-3		$(-\infty, \infty)$
Scale.y	logical	FALSE	TRUE, FALSE	-
Adjust.endspan	numeric	2		$(-\infty, \infty)$
Auto.linpreds	logical	TRUE	TRUE, FALSE	-
Force.weights	logical	FALSE	TRUE, FALSE	-
Use.beta.cache	logical	TRUE	TRUE, FALSE	-
Force.xtx.prune	logical	FALSE	TRUE, FALSE	-
Get.leverages	logical	TRUE	TRUE, FALSE	-
Exhaustive.tol	numeric	1e-10		$(-\infty, \infty)$

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrEarth`

Methods

Public methods:

- `LearnerRegrEarth$new()`
- `LearnerRegrEarth$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrEarth$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

pkopper

References

Milborrow, Stephen, Hastie, T, Tibshirani, R (2014). "Earth: multivariate adaptive regression spline models." *R package version*, **3**(7).

Friedman, H J (1991). "Multivariate adaptive regression splines." *The annals of statistics*, **19**(1), 1–67.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("earth", quietly = TRUE)) {
  learner = mlr3::lrn("regr.earth")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.extratrees`

Regression ExtraTrees Learner

Description

Calls extraTrees::extraTrees from package **extraTrees**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.extratrees")
lrn("regr.extratrees")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **extraTrees**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
nodesize	integer	1		$[1, \infty)$
numRandomCuts	integer	1		$(-\infty, \infty)$
evenCuts	logical	FALSE	TRUE, FALSE	-
numThreads	integer	1		$[1, \infty)$
quantile	logical	FALSE	TRUE, FALSE	-
subsetSizes	untyped	-		-
subsetGroups	untyped	-		-
tasks	untyped	-		-
probOfTaskCuts	numeric	-		$[0, 1]$

numRandomTaskCuts	integer	1		$[1, \infty)$
na.action	character	stop	stop, zero, fuse	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrExtraTrees`

Methods

Public methods:

- `LearnerRegrExtraTrees$new()`
- `LearnerRegrExtraTrees$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrExtraTrees$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Geurts, Pierre, Ernst, Damien, Wehenkel, Louis (2006). "Extremely randomized trees." *Machine learning*, **63**(1), 3–42.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("extraTrees", quietly = TRUE)) {
  learner = mlr3::lrn("regr.extratrees")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.fnn` *Regression Fast Nearest Neighbor Search Learner*

Description

Calls FNN::knn.reg from package **FNN**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.fnn")
lrn("regr.fnn")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **FNN**

Parameters

Id	Type	Default	Levels	Range
k	integer	1		$[1, \infty)$
algorithm	character	kd_tree	kd_tree, cover_tree, brute	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrFNN`

Methods

Public methods:

- `LearnerRegrFNN$new()`
- `LearnerRegrFNN$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrFNN$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Boltz, Sylvain, Debreuve, Eric, Barlaud, Michel (2007). "kNN-based high-dimensional Kullback-Leibler distance for tracking." In *Eighth International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'07)*, 16–16. IEEE.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("FNN", quietly = TRUE)) {
  learner = mlr3::lrn("regr.fnn")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.gam` *Regression Generalized Additive Model Learner*

Description

Generalized additive models. Calls mgcv::gam from package **mgcv**.

A gam formula specific to the task at hand is required for the `formula` parameter (see example and `?mgcv::formula.gam`). Beware, if no formula is provided, a fallback formula is used that will make the gam behave like a glm (this behavior is required for the unit tests). Only features specified in the formula will be used, superseding columns with col_roles "feature" in the task.

Calls mgcv::gam from package **mgcv**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.gam")
lrn("regr.gam")
```

Meta Information

- Task type: "regr"
- Predict Types: "response", "se"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **mgcv**

Parameters

Id	Type	Default	Levels	Range
family	character	gaussian	gaussian, poisson	-
formula	untyped	-		-
offset	untyped			-
method	character	GCV.Cp	GCV.Cp, GACV.Cp, REML, P-REML, ML, P-ML	-
optimizer	untyped	c , outer , newton		-
scale	numeric	0		$(-\infty, \infty)$
select	logical	FALSE	TRUE, FALSE	-
knots	untyped			-
sp	untyped			-
min.sp	untyped			-
H	untyped			-
gamma	numeric	1		$[1, \infty)$
paraPen	untyped			-
G	untyped			-
in.out	untyped			-

drop.unused.levels	logical	TRUE	TRUE, FALSE	-
drop.intercept	logical	FALSE	TRUE, FALSE	-
nthreads	integer	1		$[1, \infty)$
irls.reg	numeric	0		$[0, \infty)$
epsilon	numeric	1e-07		$[0, \infty)$
maxit	integer	200		$(-\infty, \infty)$
trace	logical	FALSE	TRUE, FALSE	-
mgcv.tol	numeric	1e-07		$[0, \infty)$
mgcv.half	integer	15		$[0, \infty)$
rank.tol	numeric	1.490116e-08		$[0, \infty)$
nlm	untyped	list		-
optim	untyped	list		-
newton	untyped	list		-
outerPIsteps	integer	0		$[0, \infty)$
idLinksBases	logical	TRUE	TRUE, FALSE	-
scalePenalty	logical	TRUE	TRUE, FALSE	-
efs.lspmax	integer	15		$[0, \infty)$
efs.tol	numeric	0.1		$[0, \infty)$
scale.est	character	fletcher	fletcher, pearson, deviance	-
edge.correct	logical	FALSE	TRUE, FALSE	-
block.size	integer	1000		$(-\infty, \infty)$
unconditional	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGam`

Methods

Public methods:

- `LearnerRegrGam$new()`
- `LearnerRegrGam$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrGam$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

JazzyPierrot

References

Hastie, J T, Tibshirani, J R (2017). *Generalized additive models*. Routledge.

Wood, Simon (2012). "mgcv: Mixed GAM Computation Vehicle with GCV/AIC/REML smoothness estimation."

Examples

```
# simple example
t = mlr3::tsk("mtcars")
l = mlr3::lrn("regr.gam")
l$param_set$values$formula = mpg ~ cyl + am + s(disp) + s(hp)
l$train(t)
l$model
```

---

mlr_learners_regr.gamboost

Boosted Generalized Additive Regression Learner

Description

Calls mboost::gamboost from package **mboost**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.gamboost")
lrn("regr.gamboost")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **mboost**

Parameters

Id	Type	Default	Levels
baselearner	character	bbs	bbs, bols, btree
dfbase	integer	4	
offset	numeric	NULL	
family	character	Gaussian	Gaussian, Laplace, Huber, Poisson, GammaReg, NBinomial, Hurdle, custom
custom.family	untyped	-	
nuirange	untyped	c , 0 , 100	
d	numeric	NULL	
mstop	integer	100	
nu	numeric	0.1	
risk	character	inbag	inbag, oobag, none
oobweights	untyped		
trace	logical	FALSE	TRUE, FALSE
stopintern	untyped	FALSE	
na.action	untyped	:: , stats , na.omit	

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGAMBoost`

Methods

Public methods:

- `LearnerRegrGAMBoost$new()`
- `LearnerRegrGAMBoost$clone()`

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

Usage:

```
LearnerRegrGAMBoost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mboost", quietly = TRUE)) {
  learner = mlr3::lrn("regr.gamboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.gausspr`

Regression Gaussian Process Learner

Description

Calls kernlab::gausspr from package **kernlab**.

Details

Parameters `sigma`, `degree`, `scale`, `offset` and `order` are added to make tuning `kpar` easier. If `kpar` is provided then these new parameters are ignored. If none are provided then the default "automatic" is used for `kpar`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.gausspr")
lrn("regr.gausspr")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "numeric", "integer", "logical", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **kernlab**

Parameters

Id	Type	Default	Levels	Ra
scaled	untyped	TRUE		-
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot	-
sigma	numeric	-		(–
degree	numeric	-		(–
scale	numeric	-		(–
offset	numeric	-		(–
order	numeric	-		(–
kpar	untyped	automatic		-
var	numeric	0.001		[0.
variance.model	logical	FALSE	TRUE, FALSE	-
tol	numeric	0.001		[0.
fit	logical	TRUE	TRUE, FALSE	-
na.action	untyped	na.omit		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGausspr`

Methods

Public methods:

- `LearnerRegrGausspr$new()`
- `LearnerRegrGausspr$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrGausspr$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**(9), 1–20.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("kernlab", quietly = TRUE)) {
  learner = mlr3::lrn("regr.gausspr")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.gbm` *Regression Gradient Boosting Machine Learner*

Description

Calls gbm::gbm from package **gbm**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.gbm")
lrn("regr.gbm")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **gbm**

Parameters

Id	Type	Default	Levels	Range
distribution	character	gaussian	gaussian, laplace, poisson, tdist	-
n.trees	integer	100		$[1, \infty)$
interaction.depth	integer	1		$[1, \infty)$
n.minobsinnode	integer	10		$[1, \infty)$
shrinkage	numeric	0.001		$[0, \infty)$
bag.fraction	numeric	0.5		$[0, 1]$
train.fraction	numeric	1		$[0, 1]$
cv.folds	integer	0		$(-\infty, \infty)$
keep.data	logical	FALSE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-
n.cores	integer	1		$(-\infty, \infty)$
var.monotone	untyped	-		-

Custom mlr3 defaults

- `keep_data`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: `keep_data` = `FALSE` saves memory during model fitting.
- `n.cores`:
  - Actual default: NULL
  - Adjusted default: 1
  - Reason for change: Suppressing the automatic internal parallelization if `cv.folds` > 0.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGBM`

Methods

Public methods:

- `LearnerRegrGBM$new()`
- `LearnerRegrGBM$importance()`

- `LearnerRegrGBM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrGBM$new()
```

Method `importance()`**:** The importance scores are extracted by `gbm::relative.influence()` from the model.

Usage:

```
LearnerRegrGBM$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Friedman, H J (2002). "Stochastic gradient boosting." *Computational statistics & data analysis*, **38**(4), 367–378.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("gbm", quietly = TRUE)) {
  learner = mlr3::lrn("regr.gbm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.glm` *Regression Generalized Linear Model Regression Learner*

Description

Calls stats::glm from package **stats**.

Details

For logistic regression please use `mlr_learners_classif.log_reg`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.glm")
lrn("regr.glm")
```

Meta Information

- Task type: "regr"
- Predict Types: "response", "se"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, 'stats'

Parameters

Id	Type	Default	Levels	Range
singular.ok	logical	TRUE	TRUE, FALSE	-
x	logical	FALSE	TRUE, FALSE	-
y	logical	TRUE	TRUE, FALSE	-
model	logical	TRUE	TRUE, FALSE	-
etastart	untyped	-		-
mustart	untyped	-		-
start	untyped			-
offset	untyped	-		-
family	character	gaussian	gaussian, poisson, quasipoisson, Gamma, inverse.gaussian	-
na.action	character	-	na.omit, na.pass, na.fail, na.exclude	-
link	character	-	logit, probit, cauchit, cloglog, identity, log, sqrt, 1/mu^2, inverse	-
epsilon	numeric	1e-08		$(-\infty, \infty)$
maxit	numeric	25		$(-\infty, \infty)$
trace	logical	FALSE	TRUE, FALSE	-
dispersion	untyped			-
type	character	link	response, link, terms	-

Custom mlr3 defaults

- `type`
  - Actual default: "link"
  - Adjusted default: "response"
  - Reason for change: Response scale more natural for predictions.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGlm`

Methods

Public methods:

- `LearnerRegrGlm$new()`
- `LearnerRegrGlm$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrGlm$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

salauer

References

Hosmer Jr, W D, Lemeshow, Stanley, Sturdivant, X R (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.

- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.
- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("stats", quietly = TRUE)) {
  learner = mlr3::lrn("regr.glm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
if (requireNamespace("stats", quietly = TRUE)) {
  learner = mlr3::lrn("regr.glm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_regr.glmboost

Boosted Generalized Linear Regression Learner

Description

Calls mboost::glmboost from package **mboost**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.glmboost")
lrn("regr.glmboost")
```

Meta Information

- Task type: “regr”
- Predict Types: “response”
- Feature Types: “integer”, “numeric”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3extralearners**, **mboost**

Parameters

Id	Type	Default	Levels
offset	numeric	NULL	
family	character	Gaussian	Gaussian, Laplace, Huber, Poisson, GammaReg, NBinomial, Hurdle, custom
custom.family	untyped	-	
nuirange	untyped	c , 0 , 100	
d	numeric	NULL	
center	logical	TRUE	TRUE, FALSE
mstop	integer	100	
nu	numeric	0.1	
risk	character	inbag	inbag, oobag, none
oobweights	untyped		
trace	logical	FALSE	TRUE, FALSE
stopintern	untyped	FALSE	
na.action	untyped	:: , stats , na.omit	
contrasts.arg	untyped	-	

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGLMBoost`

Methods

Public methods:

- `LearnerRegrGLMBoost$new()`
- `LearnerRegrGLMBoost$clone()`

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

Usage:

```
LearnerRegrGLMBoost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mboost", quietly = TRUE)) {
  learner = mlr3::lrn("regr.glmboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.IBk` *Regression IBk Learner*

Description

Calls RWeka::IBk from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.IBk")
lrn("regr.IBk")
```

Meta Information

- Task type: “regr”
- Predict Types: “response”
- Feature Types: “numeric”, “factor”, “ordered”, “integer”
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
I	logical	FALSE	TRUE, FALSE	-
F	logical	FALSE	TRUE, FALSE	-
K	integer	1		$[1, \infty)$
E	logical	FALSE	TRUE, FALSE	-
W	integer	0		$[0, \infty)$
X	logical	FALSE	TRUE, FALSE	-
A	untyped	weka.core.neighboursearch.LinearNNSearch		-
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrIBk`

Methods

Public methods:

- `LearnerRegrIBk$new()`
- `LearnerRegrIBk$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrIBk$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Aha, W D, Kibler, Dennis, Albert, K M (1991). "Instance-based learning algorithms." *Machine learning*, **6**(1), 37–66.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("regr.IBk")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.ksvm`

Regression Kernlab Support Vector Machine

Description

Calls kernlab::ksvm from package **kernlab**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.ksvm")
lrn("regr.ksvm")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **kernlab**

Parameters

Id	Type	Default	Levels	Range
scaled	logical	TRUE	TRUE, FALSE	-
type	character	eps-svr	eps-svr, nu-svr, eps-bsvr	-
kernel	character	rbfdot	rbfdot, polydot, vanilladot, laplacedot, besseldot, anovadot	-
C	numeric	1		$(-\infty, \infty)$
nu	numeric	0.2		$[0, \infty)$
epsilon	numeric	0.1		$(-\infty, \infty)$
cache	integer	40		$[1, \infty)$
tol	numeric	0.001		$[0, \infty)$
shrinking	logical	TRUE	TRUE, FALSE	-
sigma	numeric	-		$[0, \infty)$
degree	integer	-		$[1, \infty)$
scale	numeric	-		$[0, \infty)$
order	integer	-		$(-\infty, \infty)$
offset	numeric	-		$(-\infty, \infty)$
na.action	untyped	na.omit		-
fit	logical	TRUE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrKSVM`

Methods

Public methods:

- `LearnerRegrKSVM$new()`
- `LearnerRegrKSVM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrKSVM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

mboecker

References

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**(9), 1–20.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("kernlab", quietly = TRUE)) {
  learner = mlr3::lrn("regr.ksvm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.liblinear`

L2-Regularized Support Vector Regression Learner

Description

Calls LiblineaR::LiblineaR from package **LiblineaR**.

Details

Type of SVR depends on `type` argument:

- `type = 11` - L2-regularized L2-loss support vector regression (primal)
- `type = 12` – L2-regularized L2-loss support vector regression (dual)
- `type = 13` – L2-regularized L1-loss support vector regression (dual)

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.liblinear")
lrn("regr.liblinear")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **LiblineaR**

Parameters

Id	Type	Default	Levels	Range
type	integer	11		$[11, 13]$
cost	numeric	1		$[0, \infty)$
bias	numeric	1		$(-\infty, \infty)$
svr_eps	numeric	NULL		$[0, \infty)$
cross	integer	0		$[0, \infty)$
verbose	logical	FALSE	TRUE, FALSE	-
findC	logical	FALSE	TRUE, FALSE	-
useInitC	logical	TRUE	TRUE, FALSE	-

Custom mlr3 defaults

- `svr_eps`:
  - Actual default: `NULL`
  - Adjusted default: 0.001
  - Reason for change: `svr_eps` is type dependent and the "type" is handled by the mlr3learner. The default value is set to th default of the respective "type".

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrLiblineaR`

Methods

Public methods:

- `LearnerRegrLiblineaR$new()`
- `LearnerRegrLiblineaR$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrLiblineaR$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Fan, Rong-En, Chang, Kai-Wei, Hsieh, Cho-Jui, Wang, Xiang-Rui, Lin, Chih-Jen (2008). "LIBLINEAR: A library for large linear classification." *the Journal of machine Learning research*, **9**, 1871–1874.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("LiblineaR", quietly = TRUE)) {
  learner = mlr3::lrn("regr.liblinear")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_regr.lightgbm

Regression Light GBM Learner

Description

Calls lightgbm::lgb.train from package **lightgbm**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.lightgbm")
lrn("regr.lightgbm")
```

Meta Information

- Task type: “regr”
- Predict Types: “response”
- Feature Types: “numeric”, “integer”
- Required Packages: **mlr3**, **mlr3extralearners**, **lightgbm**

Parameters

Id	Type	Default	Levels
nrounds	integer	5	
objective	character	regression	regression, regression_l1, huber, fair, poisson, c
metric	character		, None, l1, l2, rmse, quantile, mape, huber, fair,
custom_eval	untyped		
verbose	integer	1	
record	logical	TRUE	TRUE, FALSE
eval_freq	integer	1	
init_model	untyped		
early_stopping_rounds	integer	-	
early_stopping_split	numeric	0	
callbacks	untyped	-	
reset_data	logical	FALSE	TRUE, FALSE
categorical_feature	untyped		
boosting	character	gbdt	gbdt, rf, dart, goss
linear_tree	logical	FALSE	TRUE, FALSE
num_iterations	integer	100	
learning_rate	numeric	0.1	
num_leaves	integer	31	
tree_learner	character	serial	serial, feature, data, voting
num_threads	integer	0	
device_type	character	cpu	cpu, gpu
seed	integer	-	
deterministic	logical	FALSE	TRUE, FALSE
force_col_wise	logical	FALSE	TRUE, FALSE
force_row_wise	logical	FALSE	TRUE, FALSE
histogram_pool_size	integer	-1	
max_depth	integer	-1	
min_data_in_leaf	integer	20	
min_sum_hessian_in_leaf	numeric	0.001	
bagging_fraction	numeric	1	
bagging_freq	integer	0	
bagging_seed	integer	3	
feature_fraction	numeric	1	
feature_fraction_bynode	numeric	1	
feature_fraction_seed	integer	2	
extra_trees	logical	FALSE	TRUE, FALSE
extra_seed	integer	6	
first_metric_only	logical	FALSE	TRUE, FALSE

max_delta_step	numeric	0	
lambda_l1	numeric	0	
lambda_l2	numeric	0	
linear_lambda	numeric	0	
min_gain_to_split	numeric	0	
drop_rate	numeric	0.1	
max_drop	integer	50	
skip_drop	numeric	0.5	
xgboost_dart_mode	logical	FALSE	TRUE, FALSE
uniform_drop	logical	FALSE	TRUE, FALSE
drop_seed	integer	4	
top_rate	numeric	0.2	
other_rate	numeric	0.1	
min_data_per_group	integer	100	
max_cat_threshold	integer	32	
cat_l2	numeric	10	
cat_smooth	numeric	10	
max_cat_to_onehot	integer	4	
top_k	integer	20	
monotone_constraints	untyped		
monotone_constraints_method	character	basic	basic, intermediate, advanced
monotone_penalty	numeric	0	
feature_contri	untyped		
forcedsplits_filename	untyped		
refit_decay_rate	numeric	0.9	
cegb_tradeoff	numeric	1	
cegb_penalty_split	numeric	0	
cegb_penalty_feature_lazy	untyped	-	
cegb_penalty_feature_coupled	untyped	-	
path_smooth	numeric	0	
interaction_constraints	untyped	-	
input_model	untyped		
output_model	untyped	LightGBM_model.txt	
saved_feature_importance_type	integer	0	
snapshot_freq	integer	-1	
max_bin	integer	255	
max_bin_by_feature	untyped		
min_data_in_bin	integer	3	
bin_construct_sample_cnt	integer	200000	
data_random_seed	integer	1	
is_enable_sparse	logical	TRUE	TRUE, FALSE
enable_bundle	logical	TRUE	TRUE, FALSE
use_missing	logical	TRUE	TRUE, FALSE
zero_as_missing	logical	FALSE	TRUE, FALSE
feature_pre_filter	logical	TRUE	TRUE, FALSE
pre_partition	logical	FALSE	TRUE, FALSE
two_round	logical	FALSE	TRUE, FALSE
header	logical	FALSE	TRUE, FALSE

group_column	untyped		
forcedbins_filename	untyped		
save_binary	logical	FALSE	TRUE, FALSE
boost_from_average	logical	TRUE	TRUE, FALSE
reg_sqrt	logical	FALSE	TRUE, FALSE
alpha	numeric	0.9	
fair_c	numeric	1	
poisson_max_delta_step	numeric	0.7	
tweedie_variance_power	numeric	1.5	
metric_freq	integer	1	
is_provide_training_metric	logical	FALSE	TRUE, FALSE
num_machines	integer	1	
local_listen_port	integer	12400	
time_out	integer	120	
machine_list_filename	untyped		
machines	untyped		
gpu_platform_id	integer	-1	
gpu_device_id	integer	-1	
gpu_use_dp	logical	FALSE	TRUE, FALSE
num_gpu	integer	1	
start_iteration	integer	0	
num_iteration	integer	-1	
pred_early_stop	logical	FALSE	TRUE, FALSE
pred_early_stop_freq	integer	10	
pred_early_stop_margin	numeric	10	
output_result	untyped	LightGBM_predict_result.txt	

Custom mlr3 defaults

- `num_threads`:
  - Actual default: 0L
  - Adjusted default: 1L
  - Reason for change: Prevents accidental conflicts with `future`.
- `verbose`:
  - Actual default: 1L
  - Adjusted default: -1L
  - Reason for change: Prevents accidental conflicts with mlr messaging system.

For categorical features either pre-process data by encoding columns or specify the categorical columns with the `categorical_feature` parameter. For this learner please do not prefix the categorical feature with name:. Instead of providing the data that is used for early stopping explicitly, the parameter `early_stopping_split` determines the proportion of the training data that is used for early stopping.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrLightGBM`

Methods

Public methods:

- `LearnerRegrLightGBM$new()`
- `LearnerRegrLightGBM$importance()`
- `LearnerRegrLightGBM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrLightGBM$new()
```

Method `importance()`**:** The importance scores are extracted from `lbg.importance`.

Usage:

```
LearnerRegrLightGBM$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

kapsner

References

Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei, Liu, Tie-Yan (2017). "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in neural information processing systems*, **30**.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("lightgbm", quietly = TRUE)) {
  learner = mlr3::lrn("regr.lightgbm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.M5Rules`

Regression M5Rules Learner

Description

Calls RWeka::M5Rules from package **RWeka**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.M5Rules")
lrn("regr.M5Rules")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "numeric", "factor", "ordered", "integer"
- Required Packages: **mlr3**, **mlr3extralearners**, **RWeka**

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
N	logical	FALSE	TRUE, FALSE	-
U	logical	FALSE	TRUE, FALSE	-
R	logical	FALSE	TRUE, FALSE	-
M	integer	4		$(-\infty, \infty)$
output_debug_info	logical	FALSE	TRUE, FALSE	-
do_not_check_capabilities	logical	FALSE	TRUE, FALSE	-
num_decimal_places	integer	2		$[1, \infty)$
batch_size	integer	100		$[1, \infty)$
options	untyped			-

Custom mlr3 defaults

- `output_debug_info`:
  - original id: output-debug-info
- `do_not_check_capabilities`:
  - original id: do-not-check-capabilities
- `num_decimal_places`:
  - original id: num-decimal-places
- `batch_size`:
  - original id: batch-size
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrM5Rules`

Methods

Public methods:

- `LearnerRegrM5Rules$new()`
- `LearnerRegrM5Rules$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrM5Rules$new()
```

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

Usage:

```
LearnerRegrM5Rules$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

henrifnk

References

Holmes, Geoffrey, Hall, Mark, Prank, Eibe (1999). "Generating rule sets from model trees." In *Australasian joint conference on artificial intelligence*, 1–12. Springer.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("RWeka", quietly = TRUE)) {
  learner = mlr3::lrn("regr.M5Rules")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.mars`

Regression Mars Learner

Description

Calls mda::mars from package **mda**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.mars")
lrn("regr.mars")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3extralearners**, **mda**

Parameters

Id	Type	Default	Levels	Range
degree	integer	1		$[1, \infty)$
nk	integer	-		$[1, \infty)$
penalty	numeric	2		$[0, \infty)$
thresh	numeric	0.001		$[0, \infty)$
prune	logical	TRUE	TRUE, FALSE	-
trace.mars	logical	FALSE	TRUE, FALSE	-
forward.step	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrMars`

Methods

Public methods:

- `LearnerRegrMars$new()`
- `LearnerRegrMars$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrMars$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hastie, J T, Tibshirani, J R (2017). *Generalized additive models*. Routledge.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mda", quietly = TRUE)) {
  learner = mlr3::lrn("regr.mars")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_regr.mob *Regression Model-based Recursive Partitioning Learner*

Description

Calls partykit::mob from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.mob")
lrn("regr.mob")
```

Meta Information

- Task type: "regr"
- Predict Types: "response", "se"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
rhs	untyped	-		-
fit	untyped	-		-
offset	untyped	-		-
cluster	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
bonferroni	logical	TRUE	TRUE, FALSE	-
minsize	integer	-		$[1, \infty)$
minsplit	integer	-		$[1, \infty)$
minbucket	integer	-		$[1, \infty)$
maxdepth	integer	Inf		$[0, \infty)$
mtry	integer	Inf		$[0, \infty)$
trim	numeric	0.1		$[0, \infty)$
breakties	logical	FALSE	TRUE, FALSE	-
parm	untyped	-		-
dfsplit	integer	-		$[0, \infty)$
prune	untyped	-		-
restart	logical	TRUE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-
caseweights	logical	TRUE	TRUE, FALSE	-
ytype	character	vector	vector, matrix, data.frame	-
xtype	character	matrix	vector, matrix, data.frame	-
terminal	untyped	object		-
inner	untyped	object		-
model	logical	TRUE	TRUE, FALSE	-
numsplit	character	left	left, center	-
catsplit	character	binary	binary, multiway	-
vcov	character	opg	opg, info, sandwich	-
ordinal	character	chisq	chisq, max, L2	-
nrep	integer	10000		$[0, \infty)$
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
additional	untyped	-		-
predict_fun	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrMob`

Methods

Public methods:

- `LearnerRegrMob$new()`

- `LearnerRegrMob$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrMob$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sumny

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("partykit", quietly = TRUE) && requireNamespace("sandwich", quietly = TRUE) && requireNamespa
  learner = mlr3::lrn("regr.mob")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.randomForest`

Regression Random Forest Learner

Description

Calls randomForest::randomForest from package **randomForest**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.randomForest")
lrn("regr.randomForest")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **randomForest**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
replace	logical	TRUE	TRUE, FALSE	-
strata	untyped	-		-
sampsize	untyped	-		-
nodesize	integer	5		$[1, \infty)$
maxnodes	integer	-		$[1, \infty)$
importance	character	FALSE	mse, nudepurity, none, FALSE	-
localImp	logical	FALSE	TRUE, FALSE	-
proximity	logical	FALSE	TRUE, FALSE	-
oob.prox	logical	-	TRUE, FALSE	-
norm.votes	logical	TRUE	TRUE, FALSE	-
do.trace	logical	FALSE	TRUE, FALSE	-
keep.forest	logical	TRUE	TRUE, FALSE	-
keep.inbag	logical	FALSE	TRUE, FALSE	-
predict.all	logical	FALSE	TRUE, FALSE	-
nodes	logical	FALSE	TRUE, FALSE	-

Super classes

mlr3::Learner -> mlr3::LearnerRegr -> LearnerRegrRandomForest

Methods

Public methods:

- LearnerRegrRandomForest$new()
- LearnerRegrRandomForest$importance()
- LearnerRegrRandomForest$oob_error()
- LearnerRegrRandomForest$clone()

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrRandomForest$new()
```

Method `importance()`**:** The importance scores are extracted from the slot `importance`. Parameter 'importance' must be set to either `"mse"` or `"nodepurity"`.

Usage:

```
LearnerRegrRandomForest$importance()
```

Returns: Named `numeric()`.

Method `oob_error()`**:** OOB errors are extracted from the model slot `mse`.

Usage:

```
LearnerRegrRandomForest$oob_error()
```

Returns: `numeric(1)`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

pat-s

References

Breiman, Leo (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("randomForest", quietly = TRUE)) {
  learner = mlr3::lrn("regr.randomForest")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.rfsrc`

Regression Random Forest SRC Learner

Description

Calls randomForestSRC::rfsrc from package **randomForestSRC**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.rfsrc")
lrn("regr.rfsrc")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "logical", "integer", "numeric", "factor"
- Required Packages: **mlr3**, **mlr3extralearners**, **randomForestSRC**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	1000		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
mtry.ratio	numeric	-		$[0, 1]$
nodesize	integer	15		$[1, \infty)$
nodedepth	integer	-		$[1, \infty)$
splitrule	character	mse	mse, quantile.regr, la.quantile.regr	-
nsplit	integer	10		$[0, \infty)$
importance	character	FALSE	FALSE, TRUE, none, permute, random, anti	-
block.size	integer	10		$[1, \infty)$
bootstrap	character	by.root	by.root, by.node, none, by.user	-
samptype	character	swor	swor, swr	-
samp	untyped	-		-
membership	logical	FALSE	TRUE, FALSE	-
sampsize	untyped	-		-
sampsize.ratio	numeric	-		$[0, 1]$
na.action	character	na.omit	na.omit, na.impute	-
nimpute	integer	1		$[1, \infty)$
ntime	integer	-		$[1, \infty)$
cause	integer	-		$[1, \infty)$
proximity	character	FALSE	FALSE, TRUE, inbag, oob, all	-
distance	character	FALSE	FALSE, TRUE, inbag, oob, all	-
forest.wt	character	FALSE	FALSE, TRUE, inbag, oob, all	-
xvar.wt	untyped	-		-
split.wt	untyped	-		-
forest	logical	TRUE	TRUE, FALSE	-
var.used	character	FALSE	FALSE, all.trees, by.tree	-
split.depth	character	FALSE	FALSE, all.trees, by.tree	-
seed	integer	-		$(-\infty, -1]$
do.trace	logical	FALSE	TRUE, FALSE	-
statistics	logical	FALSE	TRUE, FALSE	-
get.tree	untyped	-		-
outcome	character	train	train, test	-
ptn.count	integer	0		$[0, \infty)$
cores	integer	1		$[1, \infty)$

Custom mlr3 defaults

- `cores`:
  - Actual default: Auto-detecting the number of cores
  - Adjusted default: 1
  - Reason for change: Threading conflicts with explicit parallelization via **future**.
- `mtry`:

  - This hyperparameter can alternatively be set via the added hyperparameter `mtry.ratio` as `mtry = max(ceiling(mtry.ratio * n_features), 1)`. Note that `mtry` and `mtry.ratio` are mutually exclusive.

- `sampsize`:
  - This hyperparameter can alternatively be set via the added hyperparameter `sampsize.ratio` as `sampsize = max(ceiling(sampsize.ratio * n_obs), 1)`. Note that `sampsize` and `sampsize.ratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrRandomForestSRC`

Methods

Public methods:

- `LearnerRegrRandomForestSRC$new()`
- `LearnerRegrRandomForestSRC$importance()`
- `LearnerRegrRandomForestSRC$selected_features()`
- `LearnerRegrRandomForestSRC$oob_error()`
- `LearnerRegrRandomForestSRC$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrRandomForestSRC$new()
```

Method `importance()`**:** The importance scores are extracted from the model slot `importance`.

Usage:

```
LearnerRegrRandomForestSRC$importance()
```

Returns: Named `numeric()`.

Method `selected_features()`**:** Selected features are extracted from the model slot `var.used`.

Usage:

```
LearnerRegrRandomForestSRC$selected_features()
```

Returns: `character()`.

Method `oob_error()`**:** OOB error extracted from the model slot `err.rate`.

Usage:

```
LearnerRegrRandomForestSRC$oob_error()
```

Returns: `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Breiman, Leo (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("randomForestSRC", quietly = TRUE)) {
  learner = mlr3::lrn("regr.rfsrc")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_regr.rvm` *Regression Relevance Vector Machine Learner*

Description

Calls kernlab::rvm from package **kernlab**.

Details

Parameters `sigma`, `degree`, `scale`, `offset`, `order`, `length`, `lambda`, and `normalized` are added to make tuning `kpar` easier. If `kpar` is provided then these new parameters are ignored. If none are provided then the default "automatic" is used for `kpar`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("regr.rvm")
lrn("regr.rvm")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "numeric", "integer", "logical", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3extralearners**, **kernlab**

Parameters

Id	Type	Default	Levels
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot, stri
sigma	numeric	-	
degree	numeric	-	
scale	numeric	-	
offset	numeric	-	
order	numeric	-	
length	integer	-	
lambda	numeric	-	
normalized	logical	-	TRUE, FALSE
kpar	untyped	automatic	
alpha	untyped	5	
var	numeric	0.1	
var.fix	logical	FALSE	TRUE, FALSE
iterations	integer	100	
tol	numeric	2.220446e-16	
minmaxdiff	numeric	0.001	
verbosity	logical	FALSE	TRUE, FALSE
fit	logical	TRUE	TRUE, FALSE
na.action	untyped	na.omit	

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrRVM`

Methods

Public methods:

- `LearnerRegrRVM$new()`
- `LearnerRegrRVM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerRegrRVM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, Zeileis, Achim (2004). "kernlab-an S4 package for kernel methods in R." *Journal of statistical software*, **11**(9), 1–20.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("kernlab", quietly = TRUE)) {
  learner = mlr3::lrn("regr.rvm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.akritas`

Survival Akritas Estimator Learner

Description

Calls survivalmodels::akritas from package **survivalmodels**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.akritas")
lrn("surv.akritas")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "character", "numeric", "factor"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **distr6**

Parameters

Id	Type	Default	Levels	Range
lambda	numeric	0.5		$[0, 1]$
reverse	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvAkritas`

Methods

Public methods:

- `LearnerSurvAkritas$new()`
- `LearnerSurvAkritas$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvAkritas$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Akritas, G M (1994). "Nearest neighbor estimation of a bivariate distribution under random censoring." *The Annals of Statistics*, 1299–1327.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.akritas")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.blackboost

Gradient Boosting with Regression Trees Survival Learner

Description

Calls mboost::blackboost from package **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.blackboost")
lrn("surv.blackboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank", "lp"
- Feature Types: "integer", "numeric", "factor"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **mboost**, **pracma**

Parameters

Id	Type	Default	Levels	Range
family	character	coxph	coxph, weibull, loglog, lognormal, gehan, cindex, custom	-
custom.family	untyped	-		-
nuirange	untyped	c , 0 , 100		-
offset	untyped	-		-
center	logical	TRUE	TRUE, FALSE	-
mstop	integer	100		$[0, \infty)$
nu	numeric	0.1		$[0, 1]$
risk	character	-	inbag, oobag, none	-
stopintern	logical	FALSE	TRUE, FALSE	-
trace	logical	FALSE	TRUE, FALSE	-
oobweights	untyped	-		-
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Bonferroni	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
maxpts	integer	25000		$[1, \infty)$
abseps	numeric	0.001		$(-\infty, \infty)$
releps	numeric	0		$(-\infty, \infty)$
nmax	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0.95		$[0, 1]$
logmincriterion	numeric	-0.05129329		$(-\infty, 0]$
minsplit	integer	20		$[0, \infty)$
minbucket	integer	7		$[0, \infty)$
minprob	numeric	0.01		$[0, 1]$
stump	logical	FALSE	TRUE, FALSE	-

lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
nresample	integer	9999		$[1, \infty)$
tol	numeric	1.490116e-08		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$
mtry	integer	-		$[0, \infty)$
maxdepth	integer	-		$[0, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[1, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	TRUE	TRUE, FALSE	-
sigma	numeric	0.1		$[0, 1]$
ipcw	untyped	1		-
na.action	untyped	:: , stats , na.omit		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvBlackBoost`

Methods

Public methods:

- `LearnerSurvBlackBoost$new()`
- `LearnerSurvBlackBoost$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvBlackBoost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("mboost", quietly = TRUE) && requireNamespac
  learner = mlr3::lrn("surv.blackboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.cforest`

Survival Conditional Random Forest Learner

Description

Calls partykit::cforest from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.cforest")
lrn("surv.cforest")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **partykit**, **sandwich**, **coin**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	500		$[1, \infty)$
replace	logical	FALSE	TRUE, FALSE	-
fraction	numeric	0.632		$[0, 1]$
mtry	integer	-		$[0, \infty)$
mtryratio	numeric	-		$[0, 1]$
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
trace	logical	FALSE	TRUE, FALSE	-
offset	untyped	-		-
cluster	untyped	-		-
na.action	untyped	:: , stats , na.pass		-
scores	untyped	-		-
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Univariate	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
nmax	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0.95		$[0, 1]$
logmincriterion	numeric	-0.05129329		$(-\infty, \infty)$
minsplit	integer	20		$[1, \infty)$
minbucket	integer	7		$[1, \infty)$
minprob	numeric	0.01		$[0, 1]$
stump	logical	FALSE	TRUE, FALSE	-
lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
nresample	integer	9999		$[1, \infty)$
tol	numeric	1.490116e-08		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$
numsurrogate	logical	FALSE	TRUE, FALSE	-
maxdepth	integer	Inf		$[0, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[0, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	TRUE	TRUE, FALSE	-
saveinfo	logical	FALSE	TRUE, FALSE	-
update	logical	FALSE	TRUE, FALSE	-
splitflavour	character	ctree	ctree, exhaustive	-
maxvar	integer	-		$[1, \infty)$
OOB	logical	FALSE	TRUE, FALSE	-
simplify	logical	TRUE	TRUE, FALSE	-
scale	logical	TRUE	TRUE, FALSE	-
maxpts	integer	25000		$(-\infty, \infty)$
abseps	numeric	0.001		$[0, \infty)$

releps	numeric	0	$[0, \infty)$

Custom mlr3 defaults

- `mtry`:
  - This hyperparameter can alternatively be set via the added hyperparameter `mtryratio` as `mtry = max(ceiling(mtryratio * n_features), 1)`. Note that `mtry` and `mtryratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCForest`

Methods

Public methods:

- `LearnerSurvCForest$new()`
- `LearnerSurvCForest$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvCForest$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("partykit", quietly = TRUE) && requireNamesp
  learner = mlr3::lrn("surv.cforest")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.coxboost

Survival Cox Model with Likelihood Based Boosting Learner

Description

Calls CoxBoost::CoxBoost from package **CoxBoost**.

Details

Use LearnerSurvCoxboost and LearnerSurvCVCoxboost for Cox boosting without and with internal cross-validation of boosting step number, respectively. Tuning using the internal optimizer in LearnerSurvCVCoxboost may be more efficient when tuning `stepno` only. However, for tuning multiple hyperparameters, **mlr3tuning** and LearnerSurvCoxboost will likely give better results.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.coxboost")
lrn("surv.coxboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank", "lp"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **CoxBoost**, **pracma**

Parameters

Id	Type	Default	Levels	Range
unpen.index	untyped	-		-
standardize	logical	TRUE	TRUE, FALSE	-
stepno	integer	100		$[0, \infty)$
penalty	numeric	-		$(-\infty, \infty)$
criterion	character	pscore	pscore, score, hpscore, hscore	-
stepsize.factor	numeric	1		$(-\infty, \infty)$
sf.scheme	character	sigmoid	sigmoid, linear	-
pendistmat	untyped	-		-
connected.index	untyped	-		-
x.is.01	logical	FALSE	TRUE, FALSE	-
return.score	logical	TRUE	TRUE, FALSE	-
trace	logical	FALSE	TRUE, FALSE	-
at.step	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCoxboost`

Methods

Public methods:

- `LearnerSurvCoxboost$new()`
- `LearnerSurvCoxboost$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvCoxboost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Binder, Harald, Allignol, Arthur, Schumacher, Martin, Beyersmann, Jan (2009). "Boosting for high-dimensional time-to-event data with competing risks." *Bioinformatics*, **25**(7), 890–896.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("CoxBoost", quietly = TRUE) && requireNamesp
  learner = mlr3::lrn("surv.coxboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.coxtime

Survival Cox-Time Learner

Description

Calls survivalmodels::coxtime from package **survivalmodels**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.coxtime")
lrn("surv.coxtime")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”
- Feature Types: “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **distr6**, **reticulate**

Parameters

Id	Type	Default	Levels
frac	numeric	0	
standardize_time	logical	FALSE	TRUE, FALSE
log_duration	logical	FALSE	TRUE, FALSE
with_mean	logical	TRUE	TRUE, FALSE
with_std	logical	TRUE	TRUE, FALSE
num_nodes	untyped	c , 32, 32	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leak
device	untyped	-	
shrink	numeric	0	
optimizer	character	adam	adadelta, adagrad, adam, adamax, adamw, asgd, rmsprop, rprop, sgd, sparse_a
rho	numeric	0.9	
eps	numeric	1e-08	
lr	numeric	1	
weight_decay	numeric	0	
learning_rate	numeric	0.01	
lr_decay	numeric	0	
betas	untyped	c , 0.9 , 0.999	
amsgrad	logical	FALSE	TRUE, FALSE
lambd	numeric	1e-04	
alpha	numeric	0.75	
t0	numeric	1e+06	
momentum	numeric	0	
centered	logical	TRUE	TRUE, FALSE
etas	untyped	c , 0.5, 1.2	
step_sizes	untyped	c , 1e-06, 50	
dampening	numeric	0	
nesterov	logical	FALSE	TRUE, FALSE
batch_size	integer	256	
epochs	integer	1	
verbose	logical	TRUE	TRUE, FALSE
num_workers	integer	0	
shuffle	logical	TRUE	TRUE, FALSE
best_weights	logical	FALSE	TRUE, FALSE
early_stopping	logical	FALSE	TRUE, FALSE
min_delta	numeric	0	

patience	integer	10

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCoxtime`

Methods

Public methods:

- `LearnerSurvCoxtime$new()`
- `LearnerSurvCoxtime$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvCoxtime$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Kvamme, Håvard, Borgan Ø, Scheel I (2019). "Time-to-event prediction with neural networks and Cox regression." *arXiv preprint arXiv:1907.00825*.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.coxtime")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.ctree`

Survival Conditional Inference Tree Learner

Description

Calls partykit::ctree from package **partykit**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.ctree")
lrn("surv.ctree")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **partykit**, **coin**, **sandwich**

Parameters

Id	Type	Default	Levels	Range
teststat	character	quadratic	quadratic, maximum	-
splitstat	character	quadratic	quadratic, maximum	-
splittest	logical	FALSE	TRUE, FALSE	-
testtype	character	Bonferroni	Bonferroni, MonteCarlo, Univariate, Teststatistic	-
nmax	untyped	-		-
alpha	numeric	0.05		$[0, 1]$
mincriterion	numeric	0.95		$[0, 1]$
logmincriterion	numeric	-		$(-\infty, \infty)$
minsplit	integer	20		$[1, \infty)$
minbucket	integer	7		$[1, \infty)$
minprob	numeric	0.01		$[0, \infty)$

stump	logical	FALSE	TRUE, FALSE	-
lookahead	logical	FALSE	TRUE, FALSE	-
MIA	logical	FALSE	TRUE, FALSE	-
nresample	integer	9999		$[1, \infty)$
tol	numeric	-		$[0, \infty)$
maxsurrogate	integer	0		$[0, \infty)$
numsurrogate	logical	FALSE	TRUE, FALSE	-
mtry	integer	Inf		$[0, \infty)$
maxdepth	integer	Inf		$[0, \infty)$
maxvar	integer	-		$[1, \infty)$
multiway	logical	FALSE	TRUE, FALSE	-
splittry	integer	2		$[0, \infty)$
intersplit	logical	FALSE	TRUE, FALSE	-
majority	logical	FALSE	TRUE, FALSE	-
caseweights	logical	FALSE	TRUE, FALSE	-
applyfun	untyped	-		-
cores	integer	NULL		$(-\infty, \infty)$
saveinfo	logical	TRUE	TRUE, FALSE	-
update	logical	FALSE	TRUE, FALSE	-
splitflavour	character	ctree	ctree, exhaustive	-
offset	untyped	-		-
cluster	untyped	-		-
scores	untyped	-		-
doFit	logical	TRUE	TRUE, FALSE	-
maxpts	integer	25000		$(-\infty, \infty)$
abseps	numeric	0.001		$[0, \infty)$
releps	numeric	0		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCTree`

Methods

Public methods:

- `LearnerSurvCTree$new()`
- `LearnerSurvCTree$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvCTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

adibender

References

Hothorn T, Zeileis A (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R." *Journal of Machine Learning Research*, **16**(118), 3905-3909. http://jmlr.org/papers/v16/hothorn15a.html.

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." *Journal of Computational and Graphical Statistics*, **15**(3), 651–674. doi: 10.1198/106186006x133933, https://doi.org/10.1198/106186006x133933.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("partykit", quietly = TRUE) && requireNamesp
  learner = mlr3::lrn("surv.ctree")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.cv_coxboost`
Survival Cox Model with Cross-Validation Likelihood Based Boosting Learner

Description

Calls CoxBoost::cv.CoxBoost from package **CoxBoost**.

Details

Use LearnerSurvCoxboost and LearnerSurvCVCoxboost for Cox boosting without and with internal cross-validation of boosting step number, respectively. Tuning using the internal optimizer in LearnerSurvCVCoxboost may be more efficient when tuning `stepno` only. However, for tuning multiple hyperparameters, **mlr3tuning** and LearnerSurvCoxboost will likely give better results.

If `penalty == "optimCoxBoostPenalty"` then CoxBoost::optimCoxBoostPenalty is used to determine the penalty value to be used in CoxBoost::cv.CoxBoost.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.cv_coxboost")
lrn("surv.cv_coxboost")
```

Meta Information

- Task type: “surv”
- Predict Types: “distr”, “crank”, “lp”
- Feature Types: “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **CoxBoost**, **pracma**

Parameters

Id	Type	Default	Levels	Range
maxstepno	integer	100		$[0, \infty)$
K	integer	10		$[2, \infty)$
type	character	verweij	verweij, naive	-
folds	untyped			-
minstepno	integer	50		$[0, \infty)$
start.penalty	numeric	-		$(-\infty, \infty)$
iter.max	integer	10		$[1, \infty)$
upper.margin	numeric	0.05		$[0, 1]$
unpen.index	untyped	-		-

standardize	logical	TRUE	TRUE, FALSE	-
penalty	numeric	-		$(-\infty, \infty)$
criterion	character	pscore	pscore, score, hpscore, hscore	-
stepsize.factor	numeric	1		$(-\infty, \infty)$
sf.scheme	character	sigmoid	sigmoid, linear	-
pendistmat	untyped	-		-
connected.index	untyped	-		-
x.is.01	logical	FALSE	TRUE, FALSE	-
return.score	logical	TRUE	TRUE, FALSE	-
trace	logical	FALSE	TRUE, FALSE	-
at.step	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCVCoxboost`

Methods

Public methods:

- `LearnerSurvCVCoxboost$new()`
- `LearnerSurvCVCoxboost$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvCVCoxboost$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Binder, Harald, Allignol, Arthur, Schumacher, Martin, Beyersmann, Jan (2009). "Boosting for high-dimensional time-to-event data with competing risks." *Bioinformatics*, **25**(7), 890–896.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("CoxBoost", quietly = TRUE) && requireNamesp
  learner = mlr3::lrn("surv.cv_coxboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.deephit`

Survival DeepHit Learner

Description

Calls survivalmodels::deephit from package **survivalmodels**.

Details

Custom nets can be used in this learner either using the survivalmodels::build_pytorch_net utility function or using `torch` via **reticulate**. The number of output channels depends on the number of discretised time-points, i.e. the parameters `cuts` or `cutpoints`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.deephit")
lrn("surv.deephit")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”
- Feature Types: “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **distr6**, **reticulate**

Parameters

Id	Type	Default	Levels
frac	numeric	0	
cuts	integer	10	
cutpoints	untyped	-	
scheme	character	equidistant	equidistant, quantiles
cut_min	numeric	0	
num_nodes	untyped	c , 32, 32	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyr
custom_net	untyped	-	
device	untyped	-	
mod_alpha	numeric	0.2	
sigma	numeric	0.1	
optimizer	character	adam	adadelta, adagrad, adam, adamax, adamw, asgd, rmsprop, rprop, sgd, sparse_ada
rho	numeric	0.9	
eps	numeric	1e-08	
lr	numeric	1	
weight_decay	numeric	0	
learning_rate	numeric	0.01	
lr_decay	numeric	0	
betas	untyped	c , 0.9 , 0.999	
amsgrad	logical	FALSE	TRUE, FALSE
lambd	numeric	1e-04	
alpha	numeric	0.75	
t0	numeric	1e+06	
momentum	numeric	0	
centered	logical	TRUE	TRUE, FALSE
etas	untyped	c , 0.5, 1.2	
step_sizes	untyped	c , 1e-06, 50	
dampening	numeric	0	
nesterov	logical	FALSE	TRUE, FALSE
batch_size	integer	256	
epochs	integer	1	
verbose	logical	TRUE	TRUE, FALSE
num_workers	integer	0	
shuffle	logical	TRUE	TRUE, FALSE
best_weights	logical	FALSE	TRUE, FALSE

early_stopping	logical	FALSE	TRUE, FALSE
min_delta	numeric	0	
patience	integer	10	
interpolate	logical	FALSE	TRUE, FALSE
inter_scheme	character	const_hazard	const_hazard, const_pdf
sub	integer	10	

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvDeephit`

Methods

Public methods:

- `LearnerSurvDeephit$new()`
- `LearnerSurvDeephit$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvDeephit$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Lee, Changhee, Zame, William, Yoon, Jinsung, Van Der Schaar, Mihaela (2018). "Deephit: A deep learning approach to survival analysis with competing risks." In *Proceedings of the AAAI conference on artificial intelligence*, volume 32 number 1.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.

- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.deephit")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.deepsurv`

Survival DeepSurv Learner

Description

Calls survivalmodels::deepsurv from package **survivalmodels**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.deepsurv")
lrn("surv.deepsurv")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **distr6**, **reticulate**

Parameters

Id	Type	Default	Levels
frac	numeric	0	
num_nodes	untyped	c , 32, 32	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	

activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyr
device	untyped	-	
optimizer	character	adam	adadelta, adagrad, adam, adamax, adamw, asgd, rmsprop, rprop, sgd, sparse_ad
rho	numeric	0.9	
eps	numeric	1e-08	
lr	numeric	1	
weight_decay	numeric	0	
learning_rate	numeric	0.01	
lr_decay	numeric	0	
betas	untyped	c , 0.9 , 0.999	
amsgrad	logical	FALSE	TRUE, FALSE
lambd	numeric	1e-04	
alpha	numeric	0.75	
t0	numeric	1e+06	
momentum	numeric	0	
centered	logical	TRUE	TRUE, FALSE
etas	untyped	c , 0.5, 1.2	
step_sizes	untyped	c , 1e-06, 50	
dampening	numeric	0	
nesterov	logical	FALSE	TRUE, FALSE
batch_size	integer	256	
epochs	integer	1	
verbose	logical	TRUE	TRUE, FALSE
num_workers	integer	0	
shuffle	logical	TRUE	TRUE, FALSE
best_weights	logical	FALSE	TRUE, FALSE
early_stopping	logical	FALSE	TRUE, FALSE
min_delta	numeric	0	
patience	integer	10	

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvDeepsurv`

Methods

Public methods:

- `LearnerSurvDeepsurv$new()`
- `LearnerSurvDeepsurv$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvDeepsurv$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Katzman, L J, Shaham, Uri, Cloninger, Alexander, Bates, Jonathan, Jiang, Tingting, Kluger, Yuval (2018). "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network." *BMC medical research methodology*, **18**(1), 1–12.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.deepsurv")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.dnnsurv

Survival DNNSurv Learner

Description

Calls survivalmodels::dnnsurv from package **survivalmodels**.

Details

Custom nets can be used in this learner either using the survivalmodels::build_keras_net utility function or using **keras**. The number of output channels should be of length 1 and number of input channels is the number of features plus number of cuts.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.dnnsurv")
lrn("surv.dnnsurv")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **keras**, **pseudo**, **tensorflow**, **distr6**

Parameters

Id	Type	Default	Levels	Range
cuts	integer	5		$[1, \infty)$
cutpoints	untyped	-		-
custom_model	untyped	-		-
optimizer	character	adam	adadelta, adagrad, adamax, adam, nadam, rmsprop, sgd	-
lr	numeric	0.02		$[0, \infty)$
beta_1	numeric	0.9		$[0, 1]$
beta_2	numeric	0.999		$[0, 1]$
epsilon	numeric	-		$[0, \infty)$
decay	numeric	0		$[0, \infty)$
clipnorm	numeric	-		$(-\infty, \infty)$
clipvalue	numeric	-		$(-\infty, \infty)$
schedule_decay	numeric	0.04		$(-\infty, \infty)$
momentum	numeric	0		$[0, \infty)$
nesterov	logical	FALSE	TRUE, FALSE	-
loss_weights	untyped	-		-
weighted_metrics	untyped	-		-
early_stopping	logical	FALSE	TRUE, FALSE	-
min_delta	numeric	0		$[0, \infty)$
patience	integer	0		$[0, \infty)$
verbose	integer	0		$[0, 2]$
baseline	numeric	-		$(-\infty, \infty)$
restore_best_weights	logical	FALSE	TRUE, FALSE	-
batch_size	integer	32		$[1, \infty)$

epochs	integer	10		$[1, \infty)$
validation_split	numeric	0		$[0, 1]$
shuffle	logical	TRUE	TRUE, FALSE	-
sample_weight	untyped	-		-
initial_epoch	integer	0		$[0, \infty)$
steps_per_epoch	integer	-		$[1, \infty)$
validation_steps	integer	-		$[1, \infty)$
steps	integer	-		$[0, \infty)$
callbacks	untyped	-		-

Custom mlr3 defaults

- `verbose`:
  - Actual default: `1L`
  - Adjusted default: `0L`
  - Reason for change: Prevents plotting.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvDNNSurv`

Methods

Public methods:

- `LearnerSurvDNNSurv$new()`
- `LearnerSurvDNNSurv$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvDNNSurv$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Zhao, Lili, Feng, Dai (2019). "Dnnsurv: Deep neural networks for survival analysis using pseudo values." *arXiv preprint arXiv:1908.02337*.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.dnnsurv")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.flexible`

Survival Flexible Parametric Spline Learner

Description

Calls flexsurv::flexsurvspline from package **flexsurv**.

Details

The `distr` prediction is estimated using the fitted custom distributions from `flexsurv::flexsurvspline()` and the estimated coefficients however the prediction takes place in this package and not in **flexsurv** for a much faster and more efficient implementation.

As flexible spline models estimate the baseline hazard as the intercept, the linear predictor, `lp`, can be calculated as in the classical setting. i.e. For fitted coefficients, $\beta = (\beta_0, ..., \beta_P)$, and covariates $X^T = (X_0, ..., X_P)^T$, where X_0 is a column of 1s: $lp = \beta X$.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.flexible")
lrn("surv.flexible")
```

Meta Information

- Task type: “surv”
- Predict Types: “distr”, “crank”, “lp”
- Feature Types: “logical”, “integer”, “factor”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **flexsurv**, **pracma**

Parameters

Id	Type	Default	Levels	Range
bhazard	untyped	-		-
k	integer	0		$[0, \infty)$
knots	untyped	-		-
bknots	untyped	-		-
scale	character	hazard	hazard, odds, normal	-
timescale	character	log	log, identity	-
inits	untyped	-		-
rtrunc	untyped	-		-
fixedpars	untyped	-		-
cl	numeric	0.95		$[0, 1]$
maxiter	integer	30		$(-\infty, \infty)$
rel.tolerance	numeric	1e-09		$(-\infty, \infty)$
toler.chol	numeric	1e-10		$(-\infty, \infty)$
debug	integer	0		$[0, 1]$
outer.max	integer	10		$(-\infty, \infty)$

Custom mlr3 defaults

- k:
  - Actual default: `0`
  - Adjusted default: `1`
  - Reason for change: The default value of `0` is equivalent to, and a much less efficient implementation of, LearnerSurvParametric.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvFlexible`

Methods

Public methods:

- `LearnerSurvFlexible$new()`
- `LearnerSurvFlexible$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvFlexible$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Royston, Patrick, Parmar, KB M (2002). "Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects." *Statistics in medicine*, **21**(15), 2175–2197.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("flexsurv", quietly = TRUE) && requireNamesp
  learner = mlr3::lrn("surv.flexible")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.gamboost` *Boosted Generalized Additive Survival Learner*

Description

Calls mboost::gamboost from package **mboost**.

Calls mboost::gamboost from package **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.gamboost")
lrn("surv.gamboost")
```

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.gamboost")
lrn("surv.gamboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank", "lp"
- Feature Types: "integer", "numeric", "factor", "logical"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **mboost**, **pracma**

- Task type: "surv"
- Predict Types: "distr", "crank", "lp"
- Feature Types: "integer", "numeric", "factor", "logical"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **mboost**, **pracma**

Parameters

Id	Type	Default	Levels	Range
family	character	coxph	coxph, weibull, loglog, lognormal, gehan, cindex, custom	-
custom.family	untyped	-		-
nuirange	untyped	c , 0 , 100		-
offset	numeric	-		$(-\infty, \infty)$
center	logical	TRUE	TRUE, FALSE	-
mstop	integer	100		$[0, \infty)$
nu	numeric	0.1		$[0, 1]$
risk	character	inbag	inbag, oobag, none	-
stopintern	untyped	FALSE		-
trace	logical	FALSE	TRUE, FALSE	-
oobweights	untyped			-
baselearner	character	bbs	bbs, bols, btree	-
dfbase	integer	4		$[0, \infty)$
sigma	numeric	0.1		$[0, 1]$
ipcw	untyped	1		-
na.action	untyped	:: , stats , na.omit		-

Id	Type	Default	Levels	Range
family	character	coxph	coxph, weibull, loglog, lognormal, gehan, cindex, custom	-
custom.family	untyped	-		-
nuirange	untyped	c , 0 , 100		-
offset	numeric	-		$(-\infty, \infty)$
center	logical	TRUE	TRUE, FALSE	-
mstop	integer	100		$[0, \infty)$
nu	numeric	0.1		$[0, 1]$
risk	character	inbag	inbag, oobag, none	-
stopintern	untyped	FALSE		-
trace	logical	FALSE	TRUE, FALSE	-
oobweights	untyped			-
baselearner	character	bbs	bbs, bols, btree	-
dfbase	integer	4		$[0, \infty)$
sigma	numeric	0.1		$[0, 1]$
ipcw	untyped	1		-
na.action	untyped	:: , stats , na.omit		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvGAMBoost`

Methods

Public methods:

- `LearnerSurvGAMBoost$new()`
- `LearnerSurvGAMBoost$importance()`
- `LearnerSurvGAMBoost$selected_features()`
- `LearnerSurvGAMBoost$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvGAMBoost$new()
```

Method `importance()`**:** The importance scores are extracted with the function `mboost::varimp()` with the default arguments.

Usage:

```
LearnerSurvGAMBoost$importance()
```

Returns: Named `numeric()`.

Method `selected_features()`**:** Selected features are extracted with the function `mboost::variable.names.mboost()`, with `used.only = TRUE`.

Usage:

```
LearnerSurvGAMBoost$selected_features()
```

Returns: `character()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`

- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("mboost", quietly = TRUE) && requireNamespac
  learner = mlr3::lrn("surv.gamboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.gbm *Survival Gradient Boosting Machine Learner*

Description

Calls gbm::gbm from package **gbm**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.gbm")
lrn("surv.gbm")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "lp"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **gbm**

Parameters

Id	Type	Default	Levels	Range
distribution	character	coxph	coxph	-
n.trees	integer	100		$[1, \infty)$
cv.folds	integer	0		$[0, \infty)$
interaction.depth	integer	1		$[1, \infty)$

n.minobsinnode	integer	10		$[1, \infty)$
shrinkage	numeric	0.001		$[0, \infty)$
bag.fraction	numeric	0.5		$[0, 1]$
train.fraction	numeric	1		$[0, 1]$
keep.data	logical	TRUE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-
var.monotone	untyped	-		-
n.cores	integer	1		$(-\infty, \infty)$
single.tree	logical	FALSE	TRUE, FALSE	-

Custom mlr3 defaults

- `distribution`:
- Actual default: "bernoulli"
- Adjusted default: "coxph"
- Reason for change: This is the only distribution available for survival.
- `keep_data`:
  - Actual default: TRUE
  - Adjusted default: FALSE
  - Reason for change: `keep_data = FALSE` saves memory during model fitting.
- `n.cores`:
  - Actual default: NULL
  - Adjusted default: 1
  - Reason for change: Suppressing the automatic internal parallelization if `cv.folds` > 0.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvGBM`

Methods

Public methods:

- `LearnerSurvGBM$new()`
- `LearnerSurvGBM$importance()`
- `LearnerSurvGBM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvGBM$new()
```

Method `importance()`**:** The importance scores are extracted from the model slot `variable.importance`.

Usage:

```
LearnerSurvGBM$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Friedman, H J (2002). "Stochastic gradient boosting." *Computational statistics & data analysis*, **38**(4), 367–378.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("gbm", quietly = TRUE)) {
  learner = mlr3::lrn("surv.gbm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.glmboost`

Boosted Generalized Linear Survival Learner

Description

Calls mboost::glmboost from package **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.glmboost")
lrn("surv.glmboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank", "lp"
- Feature Types: "integer", "numeric", "factor", "logical"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **mboost**, **pracma**

Parameters

Id	Type	Default	Levels	Range
offset	numeric	-		$(-\infty, \infty)$
family	character	coxph	coxph, weibull, loglog, lognormal, gehan, cindex, custom	-
custom.family	untyped	-		-
nuirange	untyped	c , 0 , 100		-
center	logical	TRUE	TRUE, FALSE	-
mstop	integer	100		$[0, \infty)$
nu	numeric	0.1		$[0, 1]$
risk	character	inbag	inbag, oobag, none	-
oobweights	untyped			-
stopintern	logical	FALSE	TRUE, FALSE	-
trace	logical	FALSE	TRUE, FALSE	-
sigma	numeric	0.1		$[0, 1]$
ipcw	untyped	1		-
na.action	untyped	:: , stats , na.omit		-
contrasts.arg	untyped	-		-

Super classes

mlr3::Learner -> mlr3proba::LearnerSurv -> LearnerSurvGLMBoost

Methods

Public methods:

- LearnerSurvGLMBoost$new()
- LearnerSurvGLMBoost$clone()

Method new(): Creates a new instance of this R6 class. Importance is supported but fails tests as internally data is coerced to model matrix and original names can't be recovered.

Importance is supported but fails tests as internally data is coerced to model matrix and original names can't be recovered.

description Selected features are extracted with the function mboost::variable.names.mboost(), with used.only = TRUE. return character().

Usage:

```
LearnerSurvGLMBoost$new()
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Author(s)

RaphaelS1

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- as.data.table(mlr_learners) for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("mboost", quietly = TRUE) && requireNamespac
  learner = mlr3::lrn("surv.glmboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.loghaz`

Survival Logistic-Hazard Learner

Description

Calls survivalmodels::loghaz from package **survivalmodels**.

Details

Custom nets can be used in this learner either using the survivalmodels::build_pytorch_net utility function or using `torch` via **reticulate**. The number of output channels depends on the number of discretised time-points, i.e. the parameters `cuts` or `cutpoints`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.loghaz")
lrn("surv.loghaz")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **distr6**, **reticulate**

Parameters

Id	Type	Default	Levels
frac	numeric	0	
cuts	integer	10	
cutpoints	untyped	-	
scheme	character	equidistant	equidistant, quantiles

cut_min	numeric	0	
num_nodes	untyped	c , 32, 32	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyr
custom_net	untyped	-	
device	untyped	-	
optimizer	character	adam	adadelta, adagrad, adam, adamax, adamw, asgd, rmsprop, rprop, sgd, sparse_ad
rho	numeric	0.9	
eps	numeric	1e-08	
lr	numeric	1	
weight_decay	numeric	0	
learning_rate	numeric	0.01	
lr_decay	numeric	0	
betas	untyped	c , 0.9 , 0.999	
amsgrad	logical	FALSE	TRUE, FALSE
lambd	numeric	1e-04	
alpha	numeric	0.75	
t0	numeric	1e+06	
momentum	numeric	0	
centered	logical	TRUE	TRUE, FALSE
etas	untyped	c , 0.5, 1.2	
step_sizes	untyped	c , 1e-06, 50	
dampening	numeric	0	
nesterov	logical	FALSE	TRUE, FALSE
batch_size	integer	256	
epochs	integer	1	
verbose	logical	TRUE	TRUE, FALSE
num_workers	integer	0	
shuffle	logical	TRUE	TRUE, FALSE
best_weights	logical	FALSE	TRUE, FALSE
early_stopping	logical	FALSE	TRUE, FALSE
min_delta	numeric	0	
patience	integer	10	
interpolate	logical	FALSE	TRUE, FALSE
inter_scheme	character	const_hazard	const_hazard, const_pdf
sub	integer	10	

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvLogisticHazard`

Methods

Public methods:

- `LearnerSurvLogisticHazard$new()`
- `LearnerSurvLogisticHazard$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvLogisticHazard$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Gensheimer, F M, Narasimhan, BA (2018). "Simple discrete-time survival model for neural networks." *arXiv*.

Kvamme, Håvard, Borgan Ø, Scheel I (2019). "Time-to-event prediction with neural networks and Cox regression." *arXiv preprint arXiv:1907.00825*.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.loghaz")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.mboost`

Boosted Generalized Additive Survival Learner

Description

Calls mboost::mboost from package **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.mboost")
lrn("surv.mboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "crank", "lp"
- Feature Types: "integer", "numeric", "factor", "logical"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **mboost**

Parameters

Id	Type	Default	Levels	Range
family	character	coxph	coxph, weibull, loglog, lognormal, gehan, cindex, custom	-
custom.family	untyped	-		-
nuirange	untyped	c , 0 , 100		-
offset	numeric	-		$(-\infty, \infty)$
center	logical	TRUE	TRUE, FALSE	-
mstop	integer	100		$[0, \infty)$
nu	numeric	0.1		$[0, 1]$
risk	character	inbag	inbag, oobag, none	-
stopintern	logical	FALSE	TRUE, FALSE	-
trace	logical	FALSE	TRUE, FALSE	-
oobweights	untyped			-
baselearner	character	bbs	bbs, bols, btree	-
sigma	numeric	0.1		$[0, 1]$
ipcw	untyped	1		-
na.action	untyped	:: , stats , na.omit		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvMBoost`

Methods

Public methods:

- `LearnerSurvMBoost$new()`
- `LearnerSurvMBoost$importance()`
- `LearnerSurvMBoost$selected_features()`
- `LearnerSurvMBoost$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvMBoost$new()
```

Method `importance()`**:** The importance scores are extracted with the function `mboost::varimp()` with the default arguments.

Usage:

```
LearnerSurvMBoost$importance()
```

Returns: Named `numeric()`.

Method `selected_features()`**:** Selected features are extracted with the function `mboost::variable.names.mboost()`, with `used.only = TRUE`.

Usage:

```
LearnerSurvMBoost$selected_features()
```

Returns: `character()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Bühlmann, Peter, Yu, Bin (2003). "Boosting with the L 2 loss: regression and classification." *Journal of the American Statistical Association*, **98**(462), 324–339.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("mboost", quietly = TRUE)) {
  learner = mlr3::lrn("surv.mboost")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.nelson`

Survival Nelson-Aalen Estimator Learner

Description

Calls survival::survfit from package **survival**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.nelson")
lrn("surv.nelson")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survival**, **pracma**

Parameters

Empty ParamSet

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvNelson`

Methods

Public methods:

- `LearnerSurvNelson$new()`
- `LearnerSurvNelson$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvNelson$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Nelson, Wayne (1969). "Hazard plotting for incomplete failure data." *Journal of Quality Technology*, **1**(1), 27–52.

Nelson, Wayne (1972). "Theory and applications of hazard plotting for censored failure data." *Technometrics*, **14**(4), 945–966.

Aalen, Odd (1978). "Nonparametric inference for a family of counting processes." *The Annals of Statistics*, 701–726.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survival", quietly = TRUE) && requiresp
  learner = mlr3::lrn("surv.nelson")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.obliqueRSF`

Survival Oblique Random Survival Forest Learner

Description

Calls obliqueRSF::ORSF from package **obliqueRSF**.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.obliqueRSF")
lrn("surv.obliqueRSF")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "integer", "numeric", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **obliqueRSF**, **pracma**

Parameters

Id	Type	Default	Levels	Range
alpha	numeric	0.5		$(-\infty, \infty)$
ntree	integer	100		$[1, \infty)$
eval_times	untyped	-		-
min_events_to_split_node	integer	5		$[1, \infty)$
min_obs_to_split_node	integer	10		$[1, \infty)$
min_obs_in_leaf_node	integer	5		$[1, \infty)$
min_events_in_leaf_node	integer	1		$[1, \infty)$
nsplit	integer	25		$[1, \infty)$
gamma	numeric	0.5		$[1e-16, \infty)$
max_pval_to_split_node	numeric	0.5		$[0, 1]$
mtry	integer	-		$[1, \infty)$

mtry_ratio	numeric	-		$[0, 1]$
dfmax	integer	-		$[1, \infty)$
use.cv	logical	FALSE	TRUE, FALSE	-
verbose	logical	TRUE	TRUE, FALSE	-
compute_oob_predictions	logical	FALSE	TRUE, FALSE	-
random_seed	integer	-		$(-\infty, \infty)$

Custom mlr3 defaults

- `verbose`:
  - Actual default: `TRUE`
  - Adjusted default: `FALSE`
  - Reason for change: mlr3 already has it's own verbose set to `TRUE` by default
- `mtry`:
  - This hyperparameter can alternatively be set via the added hyperparameter `mtry_ratio` as `mtry = max(ceiling(mtry_ratio * n_features), 1)`. Note that `mtry` and `mtry_ratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvObliqueRSF`

Methods

Public methods:

- `LearnerSurvObliqueRSF$new()`
- `LearnerSurvObliqueRSF$oob_error()`
- `LearnerSurvObliqueRSF$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvObliqueRSF$new()
```

Method `oob_error()`**:** Integrated brier score OOB error extracted from the model slot `oob_error`. Concordance is also available.

Usage:

```
LearnerSurvObliqueRSF$oob_error()
```

Returns: `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

adibender

References

Jaeger BC, Long DL, Long DM, Sims M, Szychowski JM, Min Y, Mcclure LA, Howard G, Simon N (2019). "Oblique random survival forests." *The Annals of Applied Statistics*, **13**(3). doi: 10.1214/19aoas1261.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("obliqueRSF", quietly = TRUE) && requireName
  learner = mlr3::lrn("surv.obliqueRSF")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.parametric

Survival Fully Parametric Learner

Description

Calls survival::survreg from package **survival**.

Details

This learner allows you to choose a distribution and a model form to compose a predicted survival probability distribution.

The internal predict method is implemented in this package as our implementation is more efficient for composition to distributions than `survival::predict.survreg()`.

`lp` is predicted using the formula $lp = X\beta$ where X are the variables in the test data set and β are the fitted coefficients.

The distribution `distr` is composed using the `lp` and specifying a model form in the `type` hyperparameter. These are as follows, with respective survival functions,

- Accelerated Failure Time (`aft`)

$$S(t) = S_0(\frac{t}{exp(lp)})$$

- Proportional Hazards (`ph`)

$$S(t) = S_0(t)^{exp(lp)}$$

- Proportional Odds (`po`)

$$S(t) = \frac{S_0(t)}{exp(-lp) + (1 - exp(-lp))S_0(t)}$$

- Tobit (`tobit`)

$$S(t) = 1 - F((t - lp)/s)$$

where S_0 is the estimated baseline survival distribution (in this case with a given parametric form), lp is the predicted linear predictor, F is the cdf of a N(0, 1) distribution, and s is the fitted scale parameter.

Whilst any combination of distribution and model form is possible, this does not mean it will necessarily create a sensible or interpretable prediction. The following combinations are 'sensible':

- dist = "gaussian"; type = "tobit";
- dist = "weibull"; type = "ph";
- dist = "exponential"; type = "ph";
- dist = "weibull"; type = "aft";
- dist = "exponential"; type = "aft";
- dist = "loglogistic"; type = "aft";
- dist = "lognormal"; type = "aft";
- dist = "loglogistic"; type = "po";

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.parametric")
lrn("surv.parametric")
```

Meta Information

- Task type: "surv"
- Predict Types: "distr", "lp", "crank"
- Feature Types: "logical", "integer", "numeric", "factor"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survival**, **pracma**

Parameters

Id	Type	Default	Levels	Range
type	character	aft	aft, ph, po, tobit	-
na.action	untyped	-		-
dist	character	weibull	weibull, exponential, gaussian, lognormal, loglogistic	-
parms	untyped	-		-
init	untyped	-		-
scale	numeric	0		$[0, \infty)$
maxiter	integer	30		$(-\infty, \infty)$
rel.tolerance	numeric	1e-09		$(-\infty, \infty)$
toler.chol	numeric	1e-10		$(-\infty, \infty)$
debug	integer	0		$[0, 1]$
outer.max	integer	10		$(-\infty, \infty)$
robust	logical	FALSE	TRUE, FALSE	-
score	logical	FALSE	TRUE, FALSE	-
cluster	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvParametric`

Methods

Public methods:

- `LearnerSurvParametric$new()`
- `LearnerSurvParametric$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvParametric$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Kalbfleisch, D J, Prentice, L R (2011). *The statistical analysis of failure time data*. John Wiley & Sons.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survival", quietly = TRUE) && requireNamesp
  learner = mlr3::lrn("surv.parametric")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.pchazard

Survival PC-Hazard Learner

Description

Calls survivalmodels::pchazard from package **survivalmodels**.

Details

Custom nets can be used in this learner either using the survivalmodels::build_pytorch_net utility function or using `torch` via **reticulate**. The number of output channels depends on the number of discretised time-points, i.e. the parameters `cuts` or `cutpoints`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.pchazard")
lrn("surv.pchazard")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalmodels**, **distr6**, **reticulate**

Parameters

Id	Type	Default	Levels
frac	numeric	0	
cuts	integer	10	
cutpoints	untyped	-	
scheme	character	equidistant	equidistant, quantiles
cut_min	numeric	0	
num_nodes	untyped	c , 32, 32	
batch_norm	logical	TRUE	TRUE, FALSE
reduction	character	mean	mean, none, sum
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyr
custom_net	untyped	-	
device	untyped	-	
optimizer	character	adam	adadelta, adagrad, adam, adamax, adamw, asgd, rmsprop, rprop, sgd, sparse_ad
rho	numeric	0.9	
eps	numeric	1e-08	
lr	numeric	1	
weight_decay	numeric	0	
learning_rate	numeric	0.01	
lr_decay	numeric	0	
betas	untyped	c , 0.9 , 0.999	
amsgrad	logical	FALSE	TRUE, FALSE
lambd	numeric	1e-04	
alpha	numeric	0.75	
t0	numeric	1e+06	
momentum	numeric	0	
centered	logical	TRUE	TRUE, FALSE
etas	untyped	c , 0.5, 1.2	
step_sizes	untyped	c , 1e-06, 50	
dampening	numeric	0	

nesterov	logical	FALSE	TRUE, FALSE
batch_size	integer	256	
epochs	integer	1	
verbose	logical	TRUE	TRUE, FALSE
num_workers	integer	0	
shuffle	logical	TRUE	TRUE, FALSE
best_weights	logical	FALSE	TRUE, FALSE
early_stopping	logical	FALSE	TRUE, FALSE
min_delta	numeric	0	
patience	integer	10	
interpolate	logical	FALSE	TRUE, FALSE
sub	integer	10	

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvPCHazard`

Methods

Public methods:

- `LearnerSurvPCHazard$new()`
- `LearnerSurvPCHazard$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvPCHazard$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Kvamme, Håvard, Borgan Ø (2019). "Continuous and discrete-time survival prediction with neural networks." *arXiv preprint arXiv:1910.06724*.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalmodels", quietly = TRUE) && require
  learner = mlr3::lrn("surv.pchazard")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.penalized`

Survival L1 and L2 Penalized Regression Learner

Description

Calls penalized::penalized from package **penalized**.

Details

The `penalized` and `unpenalized` arguments in the learner are implemented slightly differently than in `penalized::penalized()`. Here, there is no parameter for `penalized` but instead it is assumed that every variable is penalized unless stated in the `unpenalized` parameter, see examples.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.penalized")
lrn("surv.penalized")
```

Meta Information

- Task type: “surv”
- Predict Types: “distr”, “crank”
- Feature Types: “integer”, “numeric”, “factor”, “logical”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **penalized**, **pracma**

Parameters

Id	Type	Default	Levels	Range
unpenalized	untyped	-		-
lambda1	untyped	0		-
lambda2	untyped	0		-
positive	logical	FALSE	TRUE, FALSE	-
fusedl	logical	FALSE	TRUE, FALSE	-
startbeta	numeric	-		$(-\infty, \infty)$
startgamma	numeric	-		$(-\infty, \infty)$
steps	integer	1		$[1, \infty)$
epsilon	numeric	1e-10		$[0, 1]$
maxiter	integer	-		$[1, \infty)$
standardize	logical	FALSE	TRUE, FALSE	-
trace	logical	TRUE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvPenalized`

Methods

Public methods:

- `LearnerSurvPenalized$new()`
- `LearnerSurvPenalized$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvPenalized$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Goeman, J J (2010). "L1 penalized estimation in the Cox proportional hazards model." *Biometrical journal*, **52**(1), 70–84.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("penalized", quietly = TRUE) && requireNames
  learner = mlr3::lrn("surv.penalized")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

---

mlr_learners_surv.rfsrc

Survival Random Forest SRC Learner

Description

Calls randomForestSRC::rfsrc from package **randomForestSRC**.

Details

`randomForestSRC::predict.rfsrc()` returns both cumulative hazard function (chf) and survival function (surv) but uses different estimators to derive these. `chf` uses a bootstrapped Nelson-Aalen estimator, (Ishwaran, 2008) whereas `surv` uses a bootstrapped Kaplan-Meier estimator. The choice of which estimator to use is given by the extra `estimator` hyper-parameter, default is `nelson`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.rfsrc")
lrn("surv.rfsrc")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "numeric", "factor"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **randomForestSRC**, **pracma**

Parameters

Id	Type	Default	Levels	Range
ntree	integer	1000		$[1, \infty)$
mtry	integer	-		$[1, \infty)$
mtry.ratio	numeric	-		$[0, 1]$
nodesize	integer	15		$[1, \infty)$
nodedepth	integer	-		$[1, \infty)$
splitrule	character	logrank	logrank, bs.gradient, logrankscore	-
nsplit	integer	10		$[0, \infty)$
importance	character	FALSE	FALSE, TRUE, none, permute, random, anti	-
block.size	integer	10		$[1, \infty)$
bootstrap	character	by.root	by.root, by.node, none, by.user	-
samptype	character	swor	swor, swr	-
samp	untyped	-		-
membership	logical	FALSE	TRUE, FALSE	-
sampsize	untyped	-		-
sampsize.ratio	numeric	-		$[0, 1]$
na.action	character	na.omit	na.omit, na.impute	-
nimpute	integer	1		$[1, \infty)$
ntime	integer	-		$[1, \infty)$
cause	integer	-		$[1, \infty)$
proximity	character	FALSE	FALSE, TRUE, inbag, oob, all	-
distance	character	FALSE	FALSE, TRUE, inbag, oob, all	-
forest.wt	character	FALSE	FALSE, TRUE, inbag, oob, all	-
xvar.wt	untyped	-		-
split.wt	untyped	-		-
forest	logical	TRUE	TRUE, FALSE	-
var.used	character	FALSE	FALSE, all.trees, by.tree	-
split.depth	character	FALSE	FALSE, all.trees, by.tree	-
seed	integer	-		$(-\infty, -1]$
do.trace	logical	FALSE	TRUE, FALSE	-
statistics	logical	FALSE	TRUE, FALSE	-

get.tree	untyped	-		-
outcome	character	train	train, test	-
ptn.count	integer	0		$[0, \infty)$
estimator	character	nelson	nelson, kaplan	-
cores	integer	1		$[1, \infty)$

Custom mlr3 defaults

- `cores`:
  - Actual default: Auto-detecting the number of cores
  - Adjusted default: 1
  - Reason for change: Threading conflicts with explicit parallelization via **future**.
- `mtry`:
  - This hyperparameter can alternatively be set via the added hyperparameter `mtry.ratio` as `mtry = max(ceiling(mtry.ratio * n_features), 1)`. Note that `mtry` and `mtry.ratio` are mutually exclusive.
- `sampsize`:
  - This hyperparameter can alternatively be set via the added hyperparameter `sampsize.ratio` as `sampsize = max(ceiling(sampsize.ratio * n_obs), 1)`. Note that `sampsize` and `sampsize.ratio` are mutually exclusive.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvRandomForestSRC`

Methods

Public methods:

- `LearnerSurvRandomForestSRC$new()`
- `LearnerSurvRandomForestSRC$importance()`
- `LearnerSurvRandomForestSRC$selected_features()`
- `LearnerSurvRandomForestSRC$oob_error()`
- `LearnerSurvRandomForestSRC$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvRandomForestSRC$new()
```

Method `importance()`**:** The importance scores are extracted from the model slot `importance`.

Usage:

```
LearnerSurvRandomForestSRC$importance()
```

Returns: Named `numeric()`.

Method `selected_features()`: Selected features are extracted from the model slot `var.used`.

Usage:

```
LearnerSurvRandomForestSRC$selected_features()
```

Returns: `character()`.

Method `oob_error()`: OOB error extracted from the model slot `err.rate`.

Usage:

```
LearnerSurvRandomForestSRC$oob_error()
```

Returns: `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS (2008). "Random survival forests." *The Annals of Applied Statistics*, **2**(3). doi: 10.1214/08aoas169, https://doi.org/10.1214/08-aoas169.

Breiman, Leo (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32. ISSN 1573-0565, doi: 10.1023/A:1010933404324.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: https://mlr3book.mlr-org.com/basics.html#learners
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("randomForestSRC", quietly = TRUE) && requir
  learner = mlr3::lrn("surv.rfsrc")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

`mlr_learners_surv.svm` *Survival Support Vector Machine Learner*

Description

Calls survivalsvm::survivalsvm from package **survivalsvm**.

Details

Four possible SVMs can be implemented, dependent on the `type` parameter. These correspond to predicting the survival time via regression (`regression`), predicting a continuous rank (`vanbelle1`, `vanbelle2`), or a hybrid of the two (`hybrid`). Whichever `type` is chosen determines how the `crank` predict type is calculated, but in any case all can be considered a valid continuous ranking.

`makediff3` is recommended when using `type = "hybrid"`.

Dictionary

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function `lrn()`:

```
mlr_learners$get("surv.svm")
lrn("surv.svm")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “response”
- Feature Types: “integer”, “numeric”, “character”, “factor”, “logical”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survivalsvm**

Parameters

Id	Type	Default	Levels	Range
type	character	regression	regression, vanbelle1, vanbelle2, hybrid	-
diff.meth	character	-	makediff1, makediff2, makediff3	-
gamma.mu	untyped	-		-
opt.meth	character	quadprog	quadprog, ipop	-
kernel	character	lin_kernel	lin_kernel, add_kernel, rbf_kernel, poly_kernel	-
kernel.pars	untyped	-		-
sgf.sv	integer	5		$[0, \infty)$
sigf	integer	7		$[0, \infty)$
maxiter	integer	20		$[0, \infty)$
margin	numeric	0.05		$[0, \infty)$
bound	numeric	10		$[0, \infty)$
eig.tol	numeric	1e-06		$[0, \infty)$

conv.tol	numeric	1e-07	$[0, \infty)$
posd.tol	numeric	1e-08	$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvSVM`

Methods

Public methods:

- `LearnerSurvSVM$new()`
- `LearnerSurvSVM$clone()`

Method `new()`**:** Creates a new instance of this R6 class.

Usage:

```
LearnerSurvSVM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

RaphaelS1

References

Van Belle, Vanya, Pelckmans, Kristiaan, Van Huffel, Sabine, Suykens, AK J (2011). "Improved performance on high-dimensional survival data by application of Survival-SVM." *Bioinformatics*, **27**(1), 87–94.

Van Belle, Vanya, Pelckmans, Kristiaan, Van Huffel, Sabine, Suykens, AK J (2011). "Support vector methods for survival analysis: a comparison between ranking and regression approaches." *Artificial intelligence in medicine*, **53**(2), 107–118.

Shivaswamy, K P, Chu, Wei, Jansche, Martin (2007). "A support vector approach to censored targets." In *Seventh IEEE international conference on data mining (ICDM 2007)*, 655–660. IEEE.

See Also

- Dictionary of Learners: mlr3::mlr_learners.
- `as.data.table(mlr_learners)` for a table of available Learners in the running session (depending on the loaded packages).
- Chapter in the mlr3book: `https://mlr3book.mlr-org.com/basics.html#learners`
- **mlr3learners** for a selection of recommended learners.
- **mlr3cluster** for unsupervised clustering learners.
- **mlr3pipelines** to combine learners with pre- and postprocessing steps.
- **mlr3tuning** for tuning of hyperparameters, **mlr3tuningspaces** for established default tuning spaces.

Examples

```
if (requireNamespace("mlr3proba", quietly = TRUE) && requireNamespace("survivalsvm", quietly = TRUE)) {
  learner = mlr3::lrn("surv.svm")
  print(learner)

  # available parameters:
  learner$param_set$ids()
}
```

Index