Package: mlr3extralearners (via r-universe)

August 22, 2024

Title Extra Learners For mlr3

Version 0.9.0

Description Extra learners for use in mlr3.

License LGPL-3

Depends R (>= 3.1.0), mlr3 (>= 0.20.2)

Imports checkmate, data.table, methods, mlr3misc (>= 0.9.4), paradox (>= 1.0.1), R6

Suggests abess, aorsf (>= 0.1.4), actuar, apcluster, BART (>= 2.9.4), C50, catboost, coin, CoxBoost, Cubist, curl, dbarts, distr6, earth, flexsurv (>= 2.3), FNN, formattable, future, gbm, glmnet, gss, jsonlite, keras (>= 2.3.0), kernlab, knitr, ks, LiblineaR, lightgbm (>= 4.4.0), lme4, locfit, logspline, mboost, mda, mgcv, mlr3cluster, mlr3learners (>= 0.4.2), mlr3proba (>= 0.6.1), mlr3pipelines, mvtnorm, nnet, np, param6, partykit, penalized, pendensity, plugdensity, pracma, prioritylasso (>= 0.3.1), pseudo, randomForest, randomPlantedForest, randomForestSRC, ranger, remotes, reticulate (>= 1.16), rpart, rsm, rvest, RWeka, sandwich, set6, sm, stats, survival, survivalmodels (>= 0.1.19), survivalsvm, tensorflow (>= 2.0.0), testthat, xgboost

Remotes binderh/CoxBoost, catboost/catboost/catboost/R-package, mlr-org/mlr3proba, RaphaelS1/survivalmodels, PlantedML/randomPlantedForest, xoopR/distr6, xoopR/param6, xoopR/set6, ropensci/aorsf

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

Roxygen list(markdown = TRUE, r6 = TRUE)

RoxygenNote 7.3.2

Config/Needs/website rmarkdown

Repository https://mlr-org.r-universe.dev

RemoteUrl https://github.com/mlr-org/mlr3extralearners

RemoteRef v0.9.0

RemoteSha 1c297f91b41fab00817ee7c92d441e7e695187eb

Contents

`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]
- Keenan Ganz <ganzkeenan1@gmail.com> [contributor]
- Byron Jaeger <bjaeger@wakehealth.edu> [contributor]
- Zezhi Wang <homura@mail.ustc.edu.cn> [contributor]
- John Zobolas <bblodfon@gmail.com> (ORCID) [contributor]
- Lukas Burk <burk@leibniz-bips.de> (ORCID) [contributor]

`create_learner` *Create a New Learner*

Description

Helper function to create a template for a learner, as well as the test and parameter test. For more details see the mlr3book.

Usage

```
create_learner(
  path = ".",
  classname,
  type,
  key = tolower(classname),
  algorithm,
  package,
  caller,
  feature_types,
  predict_types,
  properties,
  gh_name = "Unknown",
  label = toproper(algorithm)
)
```

Arguments

path	(character(1)) The path to a folder. This is where the files will be created. In case the folder is an R package, the learner file will be create in path/R and the test files will be created in path/tests/testthat. Otherwise all the files will be created in path.
classname	(character(1)) Suffix for R6 class name passed to LearnerType 'classname'.
type	(character(1)) See mlr3::mlr_reflections$task_types$type.
key	(character(1)) key for learner, if not provided defaults to the classname in all lower case. In combination with type it creates the learner's id.
algorithm	(character(1)) Brief description of the algorithm, like "Linear Model" or "Random Forest". Is used for the title of the help package and as the label (if no other label is provided).
package	(character(1)) Package from which the learner is implemented.
caller	character(1) Training function called from the upstream 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.
gh_name	(character(1)) Your GitHub handle, used to add you as the maintainer of the learner. Defaults to "Unknown".

label (character(1))
Label for the learner, default is the value of the parameter algorithm.

Examples

```
## Not run:
path = tempfile()
dir.create(path)
create_learner(
  path = path,
  classname = "Rpart",
  type = "classif",
  key = "rpart",
  algorithm = "Decision Tree",
  package = "rpart",
  caller = "rpart",
  feature_types = c("logical", "integer", "numeric", "factor", "ordered"),
  predict_types = c("response", "prob"),
  properties = c("importance", "missings", "multiclass", "twoclass", "weights"),
  gh_name = "RaphaelS1",
  label = "Regression and Partition Tree"
)

## End(Not run)
```

---

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.

mlr_learners_classif.abess *Classification Abess Learner*

Description

Adaptive best-subset selection for classification. Calls `abess::abess()` from **abess**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.abess")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
family	character	-	binomial, multinomial, ordinal	-
tune.path	character	sequence	sequence, gsection	-
tune.type	character	gic	gic, aic, bic, ebic, cv	-
normalize	integer	NULL		$(-\infty, \infty)$
support.size	untyped	NULL		-
c.max	integer	2		$[1, \infty)$
gs.range	untyped	NULL		-
lambda	numeric	0		$[0, \infty)$
always.include	untyped	NULL		-
group.index	untyped	NULL		-
init.active.set	untyped	NULL		-
splicing.type	integer	2		$[1, 2]$
max.splicing.iter	integer	20		$[1, \infty)$
screening.num	integer	NULL		$[0, \infty)$
important.search	integer	NULL		$[0, \infty)$
warm.start	logical	TRUE	TRUE, FALSE	-
nfolds	integer	5		$(-\infty, \infty)$
foldid	untyped	NULL		-
cov.update	logical	FALSE	TRUE, FALSE	-
newton	character	exact	exact, approx	-
newton.thresh	numeric	1e-06		$[0, \infty)$
max.newton.iter	integer	NULL		$[1, \infty)$
early.stop	logical	FALSE	TRUE, FALSE	-
ic.scale	numeric	1		$[0, \infty)$
num.threads	integer	0		$[0, \infty)$
seed	integer	0		$(-\infty, \infty)$

Initial parameter values

- `num.threads`: This parameter is initialized to 1 (default is 0) to avoid conflicts with the mlr3 parallelization.

Custom mlr3 parameters

- `family` - Depending on the task type, if the parameter `family` is `NULL`, it is set to `"binomial"` for binary classification tasks and to `"multinomial"` for multiclass classification problems.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifAbess`

Methods

Public methods:

- `LearnerClassifAbess$new()`
- `LearnerClassifAbess$selected_features()`
- `LearnerClassifAbess$clone()`

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

Usage:

```
LearnerClassifAbess$new()
```

Method `selected_features()`**:** Extract the name of selected features from the model by `abess::extract()`.

Usage:

```
LearnerClassifAbess$selected_features()
```

Returns: The names of selected features

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

abess-team

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

```
# Define the Learner
learner = mlr3::lrn("classif.abess")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.AdaBoostM1`

Classification AdaBoostM1 Learner

Description

Adaptive boosting algorithm for classification. Calls `RWeka::AdaBoostM1()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.AdaBoostM1")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- 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	NULL		-

Parameter changes

- `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

```
# Define the Learner
learner = mlr3::lrn("classif.AdaBoostM1")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)
```

```
# Score the predictions
predictions$score()
```

`mlr_learners_classif.bart`

Classification BART (Bayesian Additive Regression Trees) Learner

---

Description

Bayesian Additive Regression Trees are similar to gradient boosting algorithms. The classification problem is solved by 0-1 encoding of the two-class targets and setting the decision threshold to p = 0.5 during the prediction phase. Calls `dbarts::bart()` from **dbarts**.

Dictionary

This Learner can be instantiated via lrn():

```
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	FALSE	TRUE, FALSE	-
nthread	integer	1		$(-\infty, \infty)$
keepcall	logical	TRUE	TRUE, FALSE	-

sampleronly	logical	FALSE	TRUE, FALSE	-
seed	integer	NA		$(-\infty, \infty)$
proposalprobs	untyped	NULL		-
splitprobs	untyped	NULL		-
keepsampler	logical	-	TRUE, FALSE	-

Parameter Changes

- 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: 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.

Chipman, A H, George, I E, McCulloch, E R (2010). "BART: Bayesian additive regression trees." *The Annals of Applied Statistics*, **4**(1), 266–298.

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

```
# Define the Learner
learner = mlr3::lrn("classif.bart")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.bayes_net`
Classification Bayes Network Learner

Description

Bayes Network learning using various search algorithms. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `N` removed:
  - Initial structure is empty
- `P` removed:
  - Maximum number of parents
- `R` removed:
  - Random order
- `mbc` removed:
  - Applies a Markov Blanket correction to the network structure, after a network structure is learned
- `S` removed:
  - Score type
- `A` removed:
  - Initial count (alpha)
- Reason for change: The parameters are removed because they don't work out of the box and it's unclear how to use them.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.bayes_net")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
subset	untyped	-	
na.action	untyped	-	
D	logical	-	TRUE, FALSE
B	untyped	-	
Q	character	-	global.K2, global.HillClimber, global.SimulatedAnnealing, global.TabuSe
E	character	-	estimate.SimpleEstimator, estimate.BMAEstimator, estimate.MultiNomia
output_debug_info	logical	FALSE	TRUE, FALSE
do_not_check_capabilities	logical	FALSE	TRUE, FALSE
num_decimal_places	integer	2	
batch_size	integer	100	
options	untyped	NULL	

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifBayesNet`

Methods

Public methods:

- `LearnerClassifBayesNet$new()`
- `LearnerClassifBayesNet$clone()`

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

Usage:

```
LearnerClassifBayesNet$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.bayes_net")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_classif.C50

Classification C5.0 Learner

Description

Decision Tree Algorithm. Calls `C50::C5.0.formula()` from **C50**.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-
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

Gradient boosting algorithm that also supports categorical data. Calls `catboost::catboost.train()` from package 'catboost'.

Dictionary

This Learner can be instantiated via lrn():

```
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
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	NULL	
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	
early_stopping_rounds	integer	-	
eval_metric	untyped	-	
use_best_model	logical	-	TRUE, FALSE
iterations	integer	1000	

Installation

See https://catboost.ai/en/docs/concepts/r-installation.

Initial parameter values

- `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

Early stopping

Early stopping can be used to find the optimal number of boosting rounds. Set `early_stopping_rounds` to an integer value to monitor the performance of the model on the validation set while training. For information on how to configure the validation set, see the *Validation* section of `mlr3::Learner`.

Super classes

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

Active bindings

`internal_valid_scores` The last observation of the validation scores for all metrics. Extracted from `model$evaluation_log`

`internal_tuned_values` Returns the early stopped iterations if `early_stopping_rounds` was set during training.

`validate` How to construct the internal validation data. This parameter can be either `NULL`, a ratio, `"test"`, or `"predefined"`.

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

```
# Define the Learner
learner = mlr3::lrn("classif.catboost",
  iterations = 100)

print(learner)

# Define a Task
task = tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.cforest`
Classification Conditional Random Forest Learner

Description

A random forest based on conditional inference trees (ctree). Calls `partykit::cforest()` from **partykit**.

Dictionary

This Learner can be instantiated via lrn():

```
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 parameters

- `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

```
task = tsk("iris")
learner = lrn("classif.cforest", ntree = 50)
splits = partition(task)
learner$train(task, splits$train)
pred = learner$predict(task, splits$test)
```

---

mlr_learners_classif.ctree

Classification Conditional Inference Tree Learner

Description

Classification Partition Tree where a significance test is used to determine the univariate splits. Calls `partykit::ctree()` from **partykit**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("classif.ctree")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.decision_stump`
Classification Decision Stump Learner

Description

Decision Stump Learner. Calls `RWeka::DecisionStump()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.decision_stump")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
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	NULL	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifDecisionStump`

Methods

Public methods:

- `LearnerClassifDecisionStump$new()`
- `LearnerClassifDecisionStump$clone()`

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

Usage:

```
LearnerClassifDecisionStump$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.decision_stump")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.decision_table`

Classification Decision Table Learner

Description

Simple Decision Table majority classifier. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Initial parameter values

- `E`:
  - Has only 2 out of 4 original evaluation measures : acc and auc with acc being the default
  - Reason for change: this learner should only contain evaluation measures appropriate for classification tasks

Custom mlr3 parameters

- `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
- `P_best`:
  - original id: P
- `D_best`:
  - original id: D
- `N_best`:
  - original id: N
- `S_best`:
  - original id: S
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.decision_table")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
S	character	BestFirst	BestFirst, GreedyStepwise	-
X	integer	1		$(-\infty, \infty)$
E	character	acc	acc, auc	-
I	logical	-	TRUE, FALSE	-
R	logical	-	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		$[1, \infty)$
P_best	untyped	-		-
D_best	character	1	0, 1, 2	-
N_best	integer	-		$(-\infty, \infty)$

S_best	integer	1	$(-\infty, \infty)$
options	untyped	NULL	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifDecisionTable`

Methods

Public methods:

- `LearnerClassifDecisionTable$new()`
- `LearnerClassifDecisionTable$clone()`

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

Usage:

```
LearnerClassifDecisionTable$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Kohavi R (1995). "The Power of Decision Tables." In *8th European Conference on Machine Learning*, 174–189.

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

```
# Define the Learner
learner = mlr3::lrn("classif.decision_table")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.earth`
Classification MARS (Multivariate Adaptive Regression Splines) Learner

---

Description

This is an alternative implementation of MARS (Multivariate Adaptive Regression Splines). The classification problem is solved by 0-1 encoding of the two-class targets and setting the decision threshold to p = 0.5 during the prediction phase. MARS is trademarked and thus not used as the name. The name "earth" stands for "Enhanced Adaptive Regression Through Hinges".

Details

Methods for variance estimations are not yet implemented.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.earth")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
wp	untyped	NULL		-
offset	untyped	NULL		-
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	NULL		-
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

```
# Define the Learner
learner = mlr3::lrn("classif.earth")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.fnn`

Fast Nearest Neighbour Classification

Description

Fast Nearest Neighbour Classification. Calls `FNN::knn()` from **FNN**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("classif.fnn")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`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.

Formula

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.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.gam")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
formula	untyped	-		-
offset	untyped	NULL		-
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	NULL		-
sp	untyped	NULL		-
min.sp	untyped	NULL		-
H	untyped	NULL		-
gamma	numeric	1		$[1, \infty)$
paraPen	untyped	NULL		-
G	untyped	NULL		-
in.out	untyped	NULL		-
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	-
nei	untyped	-		-
ncv.threads	integer	1		$[1, \infty)$
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."

`mlr_learners_classif.gamboost`
Boosted Generalized Additive Classification Learner

Description

Fit a generalized additive classification model using a boosting algorithm. Calls `mboost::gamboost()` from **mboost**.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-
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

```
# Define the Learner
learner = mlr3::lrn("classif.gamboost")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
```

```
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.gausspr`

Classification Gaussian Process Learner

Description

Gaussian process for classification. Calls `kernlab::gausspr()` from **kernlab**. 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 lrn():

```
lrn("classif.gausspr")
```

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	untyped	TRUE		-
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot	-
sigma	numeric	-		$(-\infty,$
degree	numeric	-		$(-\infty,$

scale	numeric	-		$(-\infty,$
offset	numeric	-		$(-\infty,$
order	numeric	-		$(-\infty,$
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

```
# Define the Learner
learner = mlr3::lrn("classif.gausspr")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_classif.gbm

Gradient Boosting Classification Learner

Description

Gradient Boosting Classification Algorithm. Calls `gbm::gbm()` from **gbm**.

Dictionary

This Learner can be instantiated via lrn():

```
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	-		-

Initial parameter values

- `keep.data` is initialized to `FALSE` to save memory.
- `n.cores` is initialized to 1 to avoid conflicts with parallelization through future.

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

```
# Define the Learner
learner = mlr3::lrn("classif.gbm")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.glmboost`
Boosted Generalized Linear Classification Learner

Description

Fit a generalized linear classification model using a boosting algorithm. Calls `mboost::glmboost()` from **mboost**.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-

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

```
# Define the Learner
learner = mlr3::lrn("classif.glmboost")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.glmer`

Classification Linear Mixed Effect Learner

Description

Generalized linear model with random effects. Calls `lme4::glmer()` from **lme4**.

Initial parameter values

- `family` - Is set to `stats::binomial(link = "logit")`.

Formula

Although most mlr3 learners don't allow to specify the formula manually, and automatically set it by valling `task$formula()`, this learner allows to set the `formula` because it's core functionality depends it. This means that it might not always use all features that are available in the task. Be aware, that this can sometimes lead to unexpected error messages, because mlr3 checks the compatibility between the learner and the task on **all** available features.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.glmer")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”, “prob”
- Feature Types: “logical”, “integer”, “numeric”, “factor”
- Required Packages: **mlr3**, **lme4**

Parameters

Id	Type	Default	Levels
formula	untyped	-	
start	untyped	NULL	
verbose	integer	0	
offset	untyped	NULL	
contrasts	untyped	NULL	
optimizer	character	-	Nelde
restart_edge	logical	FALSE	TRUE
boundary.tol	numeric	1e-05	
calc.derivs	logical	TRUE	TRUE
check.nobs.vs.rankZ	character	ignore	ignore
check.nobs.vs.nlev	character	stop	ignore
check.nlev.gtreq.5	character	ignore	ignore
check.nlev.gtr.1	character	stop	ignore
check.nobs.vs.nRE	character	stop	ignore
check.rankX	character	message+drop.cols	messa
check.scaleX	character	warning	warnin
check.formula.LHS	character	stop	ignore
family	untyped	"stats::binomial(link = \"logit\")"	
nAGQ	integer	1	
mustart	untyped	-	
etastart	untyped	-	
check.conv.grad	untyped	"lme4::.makeCC(\"warning\", tol = 2e-3, relTol = NULL)"	
check.conv.singular	untyped	"lme4::.makeCC(action = \"message\", tol = formals(lme4::isSingular)$tol)"	
check.conv.hess	untyped	"lme4::.makeCC(action = \"warning\", tol = 1e-6)"	
optCtrl	untyped	list()	
tolPwrss	untyped	-	
compDev	logical	TRUE	TRUE
nAGQ0initStep	logical	TRUE	TRUE
check.response.not.const	untyped	"stop"	
newparams	untyped	NULL	
re.form	untyped	NULL	
random.only	logical	FALSE	TRUE
allow.new.levels	logical	FALSE	TRUE
na.action	untyped	"stats::na.pass"	

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifGlmer`

Methods

Public methods:

- `LearnerClassifGlmer$new()`
- `LearnerClassifGlmer$clone()`

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

Usage:

```
LearnerClassifGlmer$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sebffischer

References

Bates, M D (2010). "lme4: Mixed-effects modeling with R."

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

```
# Define the Learner and set parameter values
learner = lrn("classif.glmer",
  formula = credit_risk ~ (1 | credit_history) + job + property + telephone + savings)

# Define a Task
task = tsk("german_credit")
task$select(c("credit_history", "job", "property", "telephone", "savings"))

# Train the learner
learner$train(task)

print(learner$model)
```

`mlr_learners_classif.IBk`

Classification IBk Learner

Description

Instance based algorithm: K-nearest neighbours regression. Calls `RWeka::IBk()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.IBk")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
subset	untyped	-	
na.action	untyped	-	
weight	character	-	I, F
K	integer	1	
E	logical	FALSE	TRUE, FALSE
W	integer	0	

X	logical	FALSE	TRUE, FALSE
A	character	LinearNNSearch	BallTree, CoverTree, FilteredNeighbourSearch, KDTree, Linearl
output_debug_info	logical	FALSE	TRUE, FALSE
do_not_check_capabilities	logical	FALSE	TRUE, FALSE
num_decimal_places	integer	2	
batch_size	integer	100	
options	untyped	NULL	

Custom mlr3 parameters

- `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
- `weight`:
  - original id: I and F
- Reason for change: original `I` and `F` params are interdependent (`I` can only be `TRUE` when `F` is `FALSE` and vice versa). The easiest way to encode this is to combine `I` and `F` into one factor param.

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

```
# Define the Learner
learner = mlr3::lrn("classif.IBk")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.imbalanced_rfsrc`
Classification Imbalanced Random Forest Src Learner

Description

Imbalanced Random forest for classification between two classes. Calls `randomForestSRC::imbalanced.rfsrc()` from from **randomForestSRC**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.imbalanced_rfsrc")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
ntree	integer	3000		$[1, \infty)$
method	character	rfq	rfq, brf, standard	-
block.size	integer	10		$[1, \infty)$
fast	logical	FALSE	TRUE, FALSE	-
ratio	numeric	-		$[0, 1]$
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	-
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		$[1, \infty)$
save.memory	logical	FALSE	TRUE, FALSE	-
perf.type	character	-	gmean, misclass, brier, none	-
case.depth	logical	FALSE	TRUE, FALSE	-

Custom mlr3 parameters

- `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.
- `cores`: This value is set as the option `rf.cores` during training and is set to 1 by default.

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifImbalancedRandomForestSRC`

Methods

Public methods:

- `LearnerClassifImbalancedRandomForestSRC$new()`
- `LearnerClassifImbalancedRandomForestSRC$importance()`
- `LearnerClassifImbalancedRandomForestSRC$selected_features()`
- `LearnerClassifImbalancedRandomForestSRC$oob_error()`

- `LearnerClassifImbalancedRandomForestSRC$clone()`

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

Usage:

```
LearnerClassifImbalancedRandomForestSRC$new()
```

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

Usage:

```
LearnerClassifImbalancedRandomForestSRC$importance()
```

Returns: Named `numeric()`.

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

Usage:

```
LearnerClassifImbalancedRandomForestSRC$selected_features()
```

Returns: `character()`.

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

Usage:

```
LearnerClassifImbalancedRandomForestSRC$oob_error()
```

Returns: `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

HarutyunyanLiana

References

O'Brien R, Ishwaran H (2019). "A random forests quantile classifier for class imbalanced data." *Pattern Recognition*, **90**, 232–249. doi:10.1016/j.patcog.2019.01.036.

Chao C, Leo B (2004). "Using Random Forest to Learn Imbalanced Data." *University of California, Berkeley*.

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

```
# Define the Learner
learner = mlr3::lrn("classif.imbalanced_rfsrc")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_classif.J48

Classification J48 Learner

Description

Decision tree algorithm. Calls `RWeka::IBk()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.J48")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- 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	NULL		-

Custom mlr3 parameters

- `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

```
# Define the Learner
learner = mlr3::lrn("classif.J48")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)
```

```
# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.JRip`
Classification JRip Learner

Description

Repeated Incremental Pruning to Produce Error Reduction. Calls `RWeka::JRip()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.JRip")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- 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	NULL		-

Parameter changes

- `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

```
# Define the Learner
learner = mlr3::lrn("classif.JRip")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.kstar`

Classification KStar Learner

Description

Instance-based classifier which differs from other instance-based learners in that it uses an entropy-based distance function. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.kstar")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
B	integer	20		$(-\infty, \infty)$
E	logical	-	TRUE, FALSE	-
M	character	a	a, d, m, n	-
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifKStar`

Methods

Public methods:

- `LearnerClassifKStar$new()`
- `LearnerClassifKStar$clone()`

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

Usage:

```
LearnerClassifKStar$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Cleary JG, Trigg LE (1995). "K*: An Instance-based Learner Using an Entropic Distance Measure." In *12th International Conference on Machine Learning*, 108-114.

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

```
# Define the Learner
learner = mlr3::lrn("classif.kstar")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.ksvm`

Classification Kernlab Support Vector Machine

Description

Support vector machine for classification. Calls `kernlab::ksvm()` from **kernlab**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("classif.ksvm")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.liblinear`

LiblineaR Classification Learner

Description

L2 regularized support vector classification. Calls `LiblineaR::LiblineaR()` from **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 lrn():

```
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	NULL		-
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

```
# Define the Learner
learner = mlr3::lrn("classif.liblinear")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.lightgbm`
Classification LightGBM Learner

Description

Gradient boosting algorithm. Calls `lightgbm::lightgbm()` from **lightgbm**. The list of parameters can be found here and in the documentation of `lightgbm::lgb.train()`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.lightgbm")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
objective	character	-	binary, multiclass, multiclassova	-
eval	untyped	-		-
verbose	integer	1		$(-\infty, \infty)$
record	logical	TRUE	TRUE, FALSE	-
eval_freq	integer	1		$[1, \infty)$
callbacks	untyped	-		-
reset_data	logical	FALSE	TRUE, FALSE	-
boosting	character	gbdt	gbdt, rf, dart, goss	-
linear_tree	logical	FALSE	TRUE, FALSE	-
learning_rate	numeric	0.1		$[0, \infty)$
num_leaves	integer	31		$[1, 131072]$
tree_learner	character	serial	serial, feature, data, voting	-
num_threads	integer	0		$[0, \infty)$
device_type	character	cpu	cpu, gpu	-
seed	integer	-		$(-\infty, \infty)$
deterministic	logical	FALSE	TRUE, FALSE	-
data_sample_strategy	character	bagging	bagging, goss	-
force_col_wise	logical	FALSE	TRUE, FALSE	-
force_row_wise	logical	FALSE	TRUE, FALSE	-
histogram_pool_size	numeric	-1		$(-\infty, \infty)$
max_depth	integer	-1		$(-\infty, \infty)$
min_data_in_leaf	integer	20		$[0, \infty)$
min_sum_hessian_in_leaf	numeric	0.001		$[0, \infty)$
bagging_fraction	numeric	1		$[0, 1]$
pos_bagging_fraction	numeric	1		$[0, 1]$
neg_bagging_fraction	numeric	1		$[0, 1]$
bagging_freq	integer	0		$[0, \infty)$
bagging_seed	integer	3		$(-\infty, \infty)$
feature_fraction	numeric	1		$[0, 1]$
feature_fraction_bynode	numeric	1		$[0, 1]$
feature_fraction_seed	integer	2		$(-\infty, \infty)$
extra_trees	logical	FALSE	TRUE, FALSE	-
extra_seed	integer	6		$(-\infty, \infty)$
max_delta_step	numeric	0		$(-\infty, \infty)$
lambda_l1	numeric	0		$[0, \infty)$
lambda_l2	numeric	0		$[0, \infty)$
linear_lambda	numeric	0		$[0, \infty)$
min_gain_to_split	numeric	0		$[0, \infty)$
drop_rate	numeric	0.1		$[0, 1]$
max_drop	integer	50		$(-\infty, \infty)$
skip_drop	numeric	0.5		$[0, 1]$
xgboost_dart_mode	logical	FALSE	TRUE, FALSE	-
uniform_drop	logical	FALSE	TRUE, FALSE	-
drop_seed	integer	4		$(-\infty, \infty)$
top_rate	numeric	0.2		$[0, 1]$

other_rate	numeric	0.1		$[0, 1]$
min_data_per_group	integer	100		$[1, \infty)$
max_cat_threshold	integer	32		$[1, \infty)$
cat_l2	numeric	10		$[0, \infty)$
cat_smooth	numeric	10		$[0, \infty)$
max_cat_to_onehot	integer	4		$[1, \infty)$
top_k	integer	20		$[1, \infty)$
monotone_constraints	untyped	NULL		-
monotone_constraints_method	character	basic	basic, intermediate, advanced	-
monotone_penalty	numeric	0		$[0, \infty)$
feature_contri	untyped	NULL		-
forcedsplits_filename	untyped	""		-
refit_decay_rate	numeric	0.9		$[0, 1]$
cegb_tradeoff	numeric	1		$[0, \infty)$
cegb_penalty_split	numeric	0		$[0, \infty)$
cegb_penalty_feature_lazy	untyped	-		-
cegb_penalty_feature_coupled	untyped	-		-
path_smooth	numeric	0		$[0, \infty)$
interaction_constraints	untyped	-		-
use_quantized_grad	logical	TRUE	TRUE, FALSE	-
num_grad_quant_bins	integer	4		$(-\infty, \infty)$
quant_train_renew_leaf	logical	FALSE	TRUE, FALSE	-
stochastic_rounding	logical	TRUE	TRUE, FALSE	-
serializable	logical	TRUE	TRUE, FALSE	-
max_bin	integer	255		$[2, \infty)$
max_bin_by_feature	untyped	NULL		-
min_data_in_bin	integer	3		$[1, \infty)$
bin_construct_sample_cnt	integer	200000		$[1, \infty)$
data_random_seed	integer	1		$(-\infty, \infty)$
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	-
forcedbins_filename	untyped	""		-
is_unbalance	logical	FALSE	TRUE, FALSE	-
scale_pos_weight	numeric	1		$[0, \infty)$
sigmoid	numeric	1		$[0, \infty)$
boost_from_average	logical	TRUE	TRUE, FALSE	-
eval_at	untyped	1:5		-
multi_error_top_k	integer	1		$[1, \infty)$
auc_mu_weights	untyped	NULL		-
num_machines	integer	1		$[1, \infty)$
local_listen_port	integer	12400		$[1, \infty)$
time_out	integer	120		$[1, \infty)$
machines	untyped	""		-

gpu_platform_id	integer	-1		$(-\infty, \infty)$
gpu_device_id	integer	-1		$(-\infty, \infty)$
gpu_use_dp	logical	FALSE	TRUE, FALSE	-
num_gpu	integer	1		$[1, \infty)$
start_iteration_predict	integer	0		$(-\infty, \infty)$
num_iteration_predict	integer	-1		$(-\infty, \infty)$
pred_early_stop	logical	FALSE	TRUE, FALSE	-
pred_early_stop_freq	integer	10		$(-\infty, \infty)$
pred_early_stop_margin	numeric	10		$(-\infty, \infty)$
num_iterations	integer	100		$[1, \infty)$
early_stopping_rounds	integer	-		$[1, \infty)$
early_stopping_min_delta	numeric	-		$[0, \infty)$
first_metric_only	logical	FALSE	TRUE, FALSE	-

Initial parameter values

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

Custom mlr3 defaults

- `objective`: Depending if the task is binary / multiclass, the default is `"binary"` or `"multiclasss"`.

Custom mlr3 parameters

- `num_class`: This parameter is automatically inferred for multiclass tasks and does not have to be set.

Early Stopping and Validation

Early stopping can be used to find the optimal number of boosting rounds. Set `early_stopping_rounds` to an integer value to monitor the performance of the model on the validation set while training. For information on how to configure the validation set, see the *Validation* section of `mlr3::Learner`. The internal validation measure can be set the `eval` parameter which should be a list of mlr3::Measures, functions, or strings for the internal lightgbm measures. If `first_metric_only` = `FALSE` (default), the learner stops when any metric fails to improve.

Super classes

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

Active bindings

`internal_valid_scores` The last observation of the validation scores for all metrics. Extracted from `model$evaluation_log`

`internal_tuned_values` Returns the early stopped iterations if `early_stopping_rounds` was set during training.

`validate` How to construct the internal validation data. This parameter can be either `NULL`, a ratio, `"test"`, or `"predefined"`.

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

```
# Define the Learner
learner = mlr3::lrn("classif.lightgbm")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.LMT`

Classification Logistic Model Trees Learner

Description

Classification tree with logistic regression models at the leaves. Calls `RWeka::LMT()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.LMT")
```

Meta Information

- Task type: “classif”
- Predict Types: “response”, “prob”
- Feature Types: “integer”, “numeric”, “factor”, “ordered”
- 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	NULL		-

CUstom mlr3 parameters

- `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

```
# Define the Learner
learner = mlr3::lrn("classif.LMT")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.logistic`

Classification Logistic Regression Learner

Description

Multinomial Logistic Regression model with a ridge estimator. Calls `RWeka::Logistic()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.logistic")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
C	logical	FALSE	TRUE, FALSE	-
R	numeric	-		$(-\infty, \infty)$
M	integer	-1		$(-\infty, \infty)$
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifLogistic`

Methods

Public methods:

- `LearnerClassifLogistic$new()`
- `LearnerClassifLogistic$clone()`

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

Usage:

```
LearnerClassifLogistic$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

le Cessie, S., van Houwelingen, J.C. (1992). "Ridge Estimators in Logistic Regression." *Applied Statistics*, **41**(1), 191-201.

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

```
# Define the Learner
learner = mlr3::lrn("classif.logistic")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

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

Description

Least squares support vector machine for classification. Calls `kernlab::lssvm()` from **kernlab**. 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 lrn():

```
lrn("classif.lssvm")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
scaled	untyped	TRUE	
kernel	character	rbfdot	rbfdot, polydot, vanilladot, tanhdot, laplacedot, besseldot, anovadot, splinedot, string
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

```
# Define the Learner
learner = mlr3::lrn("classif.lssvm")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.mob`

Classification Model-based Recursive Partitioning Learner

Description

Model-based recursive partitioning algorithm. Calls `partykit::mob()` from **mob**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
library(mlr3)
logit_ = function(y, x, start = NULL, weights = NULL, offset = NULL, ...) {
  glm(y ~ 1, family = binomial, start = start, ...)
}
learner = LearnerClassifMob$new()
learner$param_set$values$rhs = "."
learner$param_set$values$fit = logit_
learner$param_set$values$additional = list(maxit = 100)
```

```
learner$feature_types = c("logical", "integer", "numeric", "factor", "ordered")
learner$properties = c("twoclass", "weights")

predict_fun = function(object, newdata, task, .type) {
  p = unname(predict(object, newdata = newdata, type = "response"))
  levs = task$levels(task$target_names)[[1L]]

  if (.type == "response") {
    ifelse(p < 0.5, levs[1L], levs[2L])
  } else {
    prob_vector_to_matrix(p, levs)
  }
}
task = tsk("breast_cancer")
learner$param_set$values$predict_fun = predict_fun
ids = partition(task)
learner$train(task, row_ids = ids$train)
learner$predict(task, row_ids = ids$test)
```

`mlr_learners_classif.multilayer_perceptron`
Classification MultilayerPerceptron Learner

Description

Classifier that uses backpropagation to learn a multi-layer perceptron. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `G` removed:
  - GUI will be opened
- Reason for change: The parameter is removed because we don't want to launch GUI.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.multilayer_perceptron")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
L	numeric	0.3		$[0, 1]$
M	numeric	0.2		$[0, 1]$
N	integer	500		$[1, \infty)$
V	numeric	0		$[0, 100]$
S	integer	0		$[0, \infty)$
E	integer	20		$[1, \infty)$
A	logical	FALSE	TRUE, FALSE	-
B	logical	FALSE	TRUE, FALSE	-
H	untyped	"a"		-
C	logical	FALSE	TRUE, FALSE	-
I	logical	FALSE	TRUE, FALSE	-
R	logical	FALSE	TRUE, FALSE	-
D	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifMultilayerPerceptron`

Methods

Public methods:

- `LearnerClassifMultilayerPerceptron$new()`
- `LearnerClassifMultilayerPerceptron$clone()`

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

Usage:

```
LearnerClassifMultilayerPerceptron$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.multilayer_perceptron")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.naive_bayes_multinomial`
Classification Multinomial Naive Bayes Learner From Weka

Description

Multinomial Naive Bayes classifier. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.naive_bayes_multinomial")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifNaiveBayesMultinomial`

Methods

Public methods:

- `LearnerClassifNaiveBayesMultinomial$new()`
- `LearnerClassifNaiveBayesMultinomial$clone()`

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

Usage:

```
LearnerClassifNaiveBayesMultinomial$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Mccallum A, Nigam K (1998). "A Comparison of Event Models for Naive Bayes Text Classification." In *AAAI-98 Workshop on 'Learning for Text Categorization'*.

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

```
# Define the Learner
learner = mlr3::lrn("classif.naive_bayes_multinomial")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_classif.naive_bayes_weka

Classification Naive Bayes Learner From Weka

Description

Naive Bayes Classifier Using Estimator Classes. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.naive_bayes_weka")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
K	logical	FALSE	TRUE, FALSE	-
D	logical	FALSE	TRUE, FALSE	-
O	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifNaiveBayesWeka`

Methods

Public methods:

- `LearnerClassifNaiveBayesWeka$new()`
- `LearnerClassifNaiveBayesWeka$clone()`

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

Usage:

```
LearnerClassifNaiveBayesWeka$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

John GH, Langley P (1995). "Estimating Continuous Distributions in Bayesian Classifiers." In *Eleventh Conference on Uncertainty in Artificial Intelligence*, 338-345.

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

```
# Define the Learner
learner = mlr3::lrn("classif.naive_bayes_weka")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
```

```
# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.OneR`
Classification OneR Learner

Description

One Rule classification algorithm that yields an extremely simple model. Calls `RWeka::OneR()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.OneR")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- 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	NULL		-

Custom mlr3 parameters

- `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

```
# Define the Learner
learner = mlr3::lrn("classif.OneR")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.PART`

Classification PART Learner

Description

Regression partition tree. Calls `RWeka::PART()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.PART")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "integer", "numeric", "factor", "ordered"
- 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	NULL		-

Custom mlr3 parameters

- `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

```
# Define the Learner
learner = mlr3::lrn("classif.PART")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.priority_lasso`
Classification Priority Lasso Learner

Description

Patient outcome prediction based on multi-omics data taking practitioners' preferences into account. Calls `prioritylasso::prioritylasso()` from **prioritylasso**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.priority_lasso")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
blocks	untyped	-		-
type.measure	character	-	class, auc	-
max.coef	untyped	NULL		-
block1.penalization	logical	TRUE	TRUE, FALSE	-
lambda.type	character	lambda.min	lambda.min, lambda.1se	-
standardize	logical	TRUE	TRUE, FALSE	-
nfolds	integer	5		$[1, \infty)$
foldid	untyped	NULL		-
cvoffset	logical	FALSE	TRUE, FALSE	-
cvoffsetnfolds	integer	10		$[1, \infty)$
return.x	logical	TRUE	TRUE, FALSE	-
handle.missingtestdata	character	-	none, omit.prediction, set.zero, impute.block	-
include.allintercepts	logical	FALSE	TRUE, FALSE	-
use.blocks	untyped	"all"		-
alignment	character	lambda	lambda, fraction	-
alpha	numeric	1		$[0, 1]$
big	numeric	9.9e+35		$(-\infty, \infty)$
devmax	numeric	0.999		$[0, 1]$
dfmax	integer	-		$[0, \infty)$
eps	numeric	1e-06		$[0, 1]$
epsnr	numeric	1e-08		$[0, 1]$
exclude	untyped	-		-
exmx	numeric	250		$(-\infty, \infty)$
fdev	numeric	1e-05		$[0, 1]$
gamma	untyped	-		-
grouped	logical	TRUE	TRUE, FALSE	-
intercept	logical	TRUE	TRUE, FALSE	-
keep	logical	FALSE	TRUE, FALSE	-
lambda	untyped	-		-
lambda.min.ratio	numeric	-		$[0, 1]$
lower.limits	untyped	-Inf		-
maxit	integer	100000		$[1, \infty)$
mnlam	integer	5		$[1, \infty)$
mxit	integer	100		$[1, \infty)$
mxitnr	integer	25		$[1, \infty)$
nlambda	integer	100		$[1, \infty)$
offset	untyped	NULL		-
parallel	logical	FALSE	TRUE, FALSE	-
penalty.factor	untyped	-		-
pmax	integer	-		$[0, \infty)$
pmin	numeric	1e-09		$[0, 1]$
prec	numeric	1e-10		$(-\infty, \infty)$
predict.gamma	numeric	gamma.1se		$(-\infty, \infty)$
relax	logical	FALSE	TRUE, FALSE	-
s	numeric	lambda.1se		$[0, 1]$

standardize.response	logical	FALSE	TRUE, FALSE	-
thresh	numeric	1e-07		$[0, \infty)$
trace.it	integer	0		$[0, 1]$
type.gaussian	character	-	covariance, naive	-
type.logistic	character	Newton	Newton, modified.Newton	-
type.multinomial	character	ungrouped	ungrouped, grouped	-
upper.limits	untyped	Inf		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifPriorityLasso`

Methods

Public methods:

- `LearnerClassifPriorityLasso$new()`
- `LearnerClassifPriorityLasso$selected_features()`
- `LearnerClassifPriorityLasso$clone()`

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

Usage:

```
LearnerClassifPriorityLasso$new()
```

Method `selected_features()`**:** Selected features, i.e. those where the coefficient is positive.

Usage:

```
LearnerClassifPriorityLasso$selected_features()
```

Returns: `character()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

HarutyunyanLiana

References

Simon K, Vindi J, Roman H, Tobias H, Anne-Laure B (2018). "Priority-Lasso: a simple hierarchical approach to the prediction of clinical outcome using multi-omics data." *BMC Bioinformatics*, **19**. doi:10.1186/s1285901823446.

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

```
# Define the Learner and set parameter values
learner = lrn("classif.priority_lasso", type.measure = "auc",
  blocks = list(bp1 = 1:4, bp2 = 5:9, bp3 = 10:28, bp4 = 29:1028))
print(learner)

# Define a Task
task = mlr3::as_task_classif(prioritylasso::pl_data, target = "pl_out")

# Train the learner
learner$train(task)

# print the model
print(learner$model)
```

`mlr_learners_classif.randomForest`

Classification Random Forest Learner

Description

Random forest for classification. Calls `randomForest::randomForest()` from **randomForest**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.randomForest")
```

Meta Information

- Task type: "classif"
- Predict Types: "response", "prob"
- Feature Types: "logical", "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	-
classwt	untyped	NULL		-
cutoff	untyped	-		-
strata	untyped	-		-
sampsize	untyped	-		-
nodesize	integer	1		$[1, \infty)$
maxnodes	integer	-		$[1, \infty)$
importance	character	FALSE	accuracy, gini, none	-
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

```
# Define the Learner
learner = mlr3::lrn("classif.randomForest")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.random_forest_weka`
Classification Random Forest Learner from Weka

Description

Class for constructing a random forest. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `store_out_of_bag_predictions`:
  - original id: store-out-of-bag-predictions
- `output_out_of_bag_complexity_statistics`:
  - original id: output-out-of-bag-complexity-statistics
- `num_slots`:
  - original id: num-slots
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern
- `attribute-importance` removed:
  - Compute and output attribute importance (mean impurity decrease method)
- Reason for change: The parameter is removed because it's unclear how to actually use it.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.random_forest_weka")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
P	numeric	100		$[0, 100]$
O	logical	FALSE	TRUE, FALSE	-
store_out_of_bag_predictions	logical	FALSE	TRUE, FALSE	-
output_out_of_bag_complexity_statistics	logical	FALSE	TRUE, FALSE	-
print	logical	FALSE	TRUE, FALSE	-
I	integer	100		$[1, \infty)$
num_slots	integer	1		$(-\infty, \infty)$
K	integer	0		$(-\infty, \infty)$
M	integer	1		$[1, \infty)$
V	numeric	0.001		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$
depth	integer	0		$[0, \infty)$
N	integer	0		$(-\infty, \infty)$
U	logical	FALSE	TRUE, FALSE	-
B	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifRandomForestWeka`

Methods

Public methods:

- `LearnerClassifRandomForestWeka$new()`
- `LearnerClassifRandomForestWeka$clone()`

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

Usage:

```
LearnerClassifRandomForestWeka$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.random_forest_weka")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
```

```
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.random_tree`
Classification Random Tree Learner

Description

Tree that considers K randomly chosen attributes at each node. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.random_tree")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
K	integer	0		$[0, \infty)$
M	integer	1		$[1, \infty)$
V	numeric	0.001		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$
depth	integer	0		$[0, \infty)$
N	integer	0		$[0, \infty)$
U	logical	FALSE	TRUE, FALSE	-
B	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifRandomTree`

Methods

Public methods:

- `LearnerClassifRandomTree$new()`
- `LearnerClassifRandomTree$clone()`

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

Usage:

```
LearnerClassifRandomTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.random_tree")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_classif.reptree

Classification Decision Tree Learner

Description

Fast decision tree learner. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.reptree")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
M	integer	2		$(-\infty, \infty)$
V	numeric	0.001		$(-\infty, \infty)$
N	integer	3		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$
P	logical	-	TRUE, FALSE	-
L	integer	-1		$(-\infty, \infty)$
I	integer	0		$(-\infty, \infty)$
R	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifREPTree`

Methods

Public methods:

- `LearnerClassifREPTree$new()`
- `LearnerClassifREPTree$clone()`

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

Usage:

```
LearnerClassifREPTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.reptree")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)
```

```
# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.rfsrc`

Classification Random Forest SRC Learner

Description

Random forest for classification. Calls `randomForestSRC::rfsrc()` from **randomForestSRC**.

Dictionary

This Learner can be instantiated via lrn():

```
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		$[1, \infty)$
save.memory	logical	FALSE	TRUE, FALSE	-
perf.type	character	-	gmean, misclass, brier, none	-
case.depth	logical	FALSE	TRUE, FALSE	-

Custom mlr3 parameters

- `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.
- `cores`: This value is set as the option `rf.cores` during training and is set to 1 by default.

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

```
# Define the Learner
learner = mlr3::lrn("classif.rfsrc")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_classif.rpf

Classification Random Planted Forest Learner

Description

Random Planted Forest: A directly interpretable tree ensemble.

Calls `randomPlantedForest::rpf()` from 'randomPlantedForest'.

Initial parameter values

- `loss`:
  - Actual default: `"L2"`.
  - Initial value: `"exponential"`.
  - Reason for change: Using `"L2"` (or `"L1"`) loss does not guarantee predictions are valid probabilities and more akin to the linear predictor of a GLM.

Custom mlr3 parameters

- `max_interaction`:
  - This hyperparameter can alternatively be set via `max_interaction_ratio` as `max_interaction = max(ceiling(max_interaction_ratio * n_features), 1)`. The parameter `max_interaction_limit` can optionally be set as an upper bound, such that `max_interaction_ratio * min(n_features, max_interaction_limit)` is used instead. This is analogous to `mtry.ratio` in `classif.ranger`, with `max_interaction_limit` as an additional constraint. The parameter `max_interaction_limit` is initialized to `Inf`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.rpf")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
max_interaction	integer	1		$[0, \infty)$
max_interaction_ratio	numeric	-		$[0, 1]$
max_interaction_limit	integer	-		$[1, \infty)$
ntrees	integer	50		$[1, \infty)$
splits	integer	30		$[1, \infty)$
split_try	integer	10		$[1, \infty)$
t_try	numeric	0.4		$[0, 1]$
loss	character	L2	L1, L2, logit, exponential	-
delta	numeric	1		$[0, 1]$
epsilon	numeric	0.1		$[0, 1]$
deterministic	logical	FALSE	TRUE, FALSE	-
nthreads	integer	1		$[1, \infty)$
cv	logical	FALSE	TRUE, FALSE	-

purify	logical	FALSE	TRUE, FALSE	-

Installation

Package 'randomPlantedForest' is not on CRAN and has to be installed from GitHub via `remotes::install_github("Plan`

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifRandomPlantedForest`

Methods

Public methods:

- `LearnerClassifRandomPlantedForest$new()`
- `LearnerClassifRandomPlantedForest$clone()`

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

Usage:

```
LearnerClassifRandomPlantedForest$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

jemus42

References

Hiabu, Munir, Mammen, Enno, Meyer, T. J (2023). "Random Planted Forest: a directly interpretable tree ensemble." *arXiv preprint arXiv:2012.14563*. doi:10.48550/ARXIV.2012.14563.

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

```
# Define the Learner
learner = mlr3::lrn("classif.rpf")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.sgd`
Classification Stochastic Gradient Descent Learner

Description

Stochastic Gradient Descent for learning various linear models. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Initial parameter values

- `F`:
  - Has only 2 out of 5 original loss functions: 0 = hinge loss (SVM) and 1 = log loss (logistic regression) with 0 (hinge loss) still being the default
  - Reason for change: this learner should only contain loss functions appropriate for classification tasks

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.sgd")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
F	character	0	0, 1	-
L	numeric	0.01		$(-\infty, \infty)$
R	numeric	1e-04		$(-\infty, \infty)$
E	integer	500		$(-\infty, \infty)$
C	numeric	0.001		$(-\infty, \infty)$
N	logical	-	TRUE, FALSE	-
M	logical	-	TRUE, FALSE	-
S	integer	1		$(-\infty, \infty)$
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifSGD`

Methods

Public methods:

- `LearnerClassifSGD$new()`
- `LearnerClassifSGD$clone()`

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

Usage:

```
LearnerClassifSGD$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("classif.sgd")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.simple_logistic`
Classification SimpleLogistic Learner

Description

LogitBoost with simple regression functions as base learners. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.simple_logistic")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
I	integer	-		$(-\infty, \infty)$
S	logical	FALSE	TRUE, FALSE	-
P	logical	FALSE	TRUE, FALSE	-
M	integer	-		$(-\infty, \infty)$
H	integer	50		$(-\infty, \infty)$
W	numeric	0		$(-\infty, \infty)$
A	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifSimpleLogistic`

Methods

Public methods:

- `LearnerClassifSimpleLogistic$new()`
- `LearnerClassifSimpleLogistic$clone()`

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

Usage:

```
LearnerClassifSimpleLogistic$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

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

Sumner M, Frank E, Hall M (2005). "Speeding up Logistic Model Tree Induction." In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 675-683.

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

```
# Define the Learner
learner = mlr3::lrn("classif.simple_logistic")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.smo`
Classification Support Vector Machine Learner

Description

Support Vector classifier trained with John Platt's sequential minimal optimization algorithm. Calls `RWeka::SMO()` from **RWeka**.

Custom mlr3 parameters

- `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
- `E_poly`:
  - original id: E
- `L_poly`:
  - original id: L
- `C_poly`:
  - original id: C
- `C_logistic`:
  - original id: C
- `R_logistic`:
  - original id: R
- `M_logistic`:
  - original id: M
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.smo")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
subset	untyped	-	
na.action	untyped	-	
no_checks	logical	FALSE	TRUE, FALSE
C	numeric	1	
N	character	0	0, 1, 2
L	numeric	0.001	
P	numeric	1e-12	
M	logical	FALSE	TRUE, FALSE
V	integer	-1	
W	integer	1	
K	character	PolyKernel	NormalizedPolyKernel, PolyKernel, Puk, RB
calibrator	untyped	"weka.classifiers.functions.Logistic"	
E_poly	numeric	1	
L_poly	logical	FALSE	TRUE, FALSE
C_poly	integer	25007	
C_logistic	logical	FALSE	TRUE, FALSE
R_logistic	numeric	-	
M_logistic	integer	-1	
output_debug_info	logical	FALSE	TRUE, FALSE
do_not_check_capabilities	logical	FALSE	TRUE, FALSE
num_decimal_places	integer	2	
batch_size	integer	100	
options	untyped	NULL	

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifSMO`

Methods

Public methods:

- `LearnerClassifSMO$new()`
- `LearnerClassifSMO$clone()`

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

Usage:

```
LearnerClassifSMO$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Platt J (1998). "Fast Training of Support Vector Machines using Sequential Minimal Optimization." In Schoelkopf B, Burges C, Smola A (eds.), *Advances in Kernel Methods - Support Vector Learning*. MIT Press. http://research.microsoft.com/jplatt/smo.html.

Keerthi S, Shevade S, Bhattacharyya C, Murthy K (2001). "Improvements to Platt's SMO Algorithm for SVM Classifier Design." *Neural Computation*, **13**(3), 637-649.

Hastie T, Tibshirani R (1998). "Classification by Pairwise Coupling." In Jordan MI, Kearns MJ, Solla SA (eds.), *Advances in Neural Information Processing Systems*, volume 10.

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

```
# Define the Learner
learner = mlr3::lrn("classif.smo")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)
```

```
# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_classif.voted_perceptron`
Classification Voted Perceptron Learner

Description

Voted Perceptron Algorithm by Freund and Schapire. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("classif.voted_perceptron")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
I	integer	1		$[1, \infty)$
E	numeric	1		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$
M	integer	10000		$(-\infty, \infty)$
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerClassif` -> `LearnerClassifVotedPerceptron`

Methods

Public methods:

- `LearnerClassifVotedPerceptron$new()`
- `LearnerClassifVotedPerceptron$clone()`

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

Usage:

```
LearnerClassifVotedPerceptron$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Freund Y, Schapire RE (1998). "Large margin classification using the perceptron algorithm." In *11th Annual Conference on Computational Learning Theory*, 209-217.

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

```
# Define the Learner
learner = mlr3::lrn("classif.voted_perceptron")
print(learner)

# Define a Task
task = mlr3::tsk("sonar")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_dens.kde_ks

Density KS Kernel Learner

Description

Kernel density estimator. Calls `ks::kde()` from **ks**.

Dictionary

This Learner can be instantiated via lrn():

```
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	-
density	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

```
# Define the Learner
learner = mlr3::lrn("dens.kde_ks")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
```

```
# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_dens.locfit`
Density Locfit Learner

Description

Local density estimation. Calls `locfit::density.lf()` from **locfit**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.locfit")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_dens.logspline`

Density Logspline Learner

Description

Density logspline learner. Calls `logspline::logspline()` from **logspline**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.logspline")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_dens.mixed`

Density Mixed Data Kernel Learner

Description

Density estimator for discrete and continuous variables. Calls `np::npudens()` from **np**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.mixed")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_dens.nonpar`

Density Nonparametric Learner

Description

Nonparametric density estimation. Calls `sm::sm.density()` from **sm**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.nonpar")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_dens.pen *Density Penalized Learner*

Description

Density estimation using penalized B-splines with automatic selection of smoothing parameter. Calls `pendensity::pendensity()` from **pendensity**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.pen")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_dens.plug`

Density Plug-In Kernel Learner

Description

Kernel density estimation with global bandwidth selection via "plug-in". Calls `plugdensity::plugin.density()` from **plugdensity**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.plug")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_dens.spline`
Density Smoothing Splines Learner

Description

Density Smoothing Splines Learner. Calls `gss::ssden()` from **gss**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("dens.spline")
print(learner)

# Define a Task
task = mlr3::tsk("faithful")

# Create train and test set
```

```
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.abess`

Regression Abess Learner

Description

Adaptive best-subset selection for regression. Calls `abess::abess()` from **abess**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.abess")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
family	character	gaussian	gaussian, mgaussian, poisson, gamma	-
tune.path	character	sequence	sequence, gsection	-
tune.type	character	gic	gic, aic, bic, ebic, cv	-
normalize	integer	NULL		$(-\infty, \infty)$
support.size	untyped	NULL		-
c.max	integer	2		$[1, \infty)$
gs.range	untyped	NULL		-

lambda	numeric	0		$[0, \infty)$
always.include	untyped	NULL		-
group.index	untyped	NULL		-
init.active.set	untyped	NULL		-
splicing.type	integer	2		$[1, 2]$
max.splicing.iter	integer	20		$[1, \infty)$
screening.num	integer	NULL		$[0, \infty)$
important.search	integer	NULL		$[0, \infty)$
warm.start	logical	TRUE	TRUE, FALSE	-
nfolds	integer	5		$(-\infty, \infty)$
foldid	untyped	NULL		-
cov.update	logical	FALSE	TRUE, FALSE	-
newton	character	exact	exact, approx	-
newton.thresh	numeric	1e-06		$[0, \infty)$
max.newton.iter	integer	NULL		$[1, \infty)$
early.stop	logical	FALSE	TRUE, FALSE	-
ic.scale	numeric	1		$[0, \infty)$
num.threads	integer	0		$[0, \infty)$
seed	integer	1		$(-\infty, \infty)$

Initial parameter values

- `num.threads`: This parameter is initialized to 1 (default is 0) to avoid conflicts with the mlr3 parallelization.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrAbess`

Methods

Public methods:

- `LearnerRegrAbess$new()`
- `LearnerRegrAbess$selected_features()`
- `LearnerRegrAbess$clone()`

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

Usage:

```
LearnerRegrAbess$new()
```

Method `selected_features()`**:** Extract the name of selected features from the model by `abess::extract()`.

Usage:

```
LearnerRegrAbess$selected_features()
```

Returns: The names of selected features

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

abess-team

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

```
# Define the Learner
learner = mlr3::lrn("regr.abess")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.bart`

Regression BART (Bayesian Additive Regression Trees) Learner

Description

Bayesian Additive Regression Trees are similar to gradient boosting algorithms. Calls `dbarts::bart()` from **dbarts**.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-
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	FALSE	TRUE, FALSE	-
nthread	integer	1		$(-\infty, \infty)$
keeptrees	logical	FALSE	TRUE, FALSE	-
keepcall	logical	TRUE	TRUE, FALSE	-
sampleronly	logical	FALSE	TRUE, FALSE	-

seed	integer	NA		$(-\infty, \infty)$
proposalprobs	untyped	NULL		-
splitprobs	untyped	NULL		-
keepsampler	logical	-	TRUE, FALSE	-

Custom mlr3 parameters

- 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: nchain, combineChains, combinechains
  - The parameters are removed as parallelization of multiple models is handled by future.

Initial parameter values

- `keeptrees` is initialized to `TRUE` because it is required for prediction.

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.

Chipman, A H, George, I E, McCulloch, E R (2010). "BART: Bayesian additive regression trees." *The Annals of Applied Statistics*, **4**(1), 266–298.

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

```
# Define the Learner
learner = mlr3::lrn("regr.bart")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.catboost`

Gradient Boosted Decision Trees Regression Learner

Description

Gradient boosting algorithm that also supports categorical data. Calls `catboost::catboost.train()` from package 'catboost'.

Dictionary

This Learner can be instantiated via lrn():

```
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,
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	
early_stopping_rounds	integer	-	
eval_metric	untyped	-	
use_best_model	logical	-	TRUE, FALSE
iterations	integer	1000	

Installation

See https://catboost.ai/en/docs/concepts/r-installation.

Initial parameter values

- 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

Early stopping

Early stopping can be used to find the optimal number of boosting rounds. Set early_stopping_rounds to an integer value to monitor the performance of the model on the validation set while training. For information on how to configure the validation set, see the *Validation* section of mlr3::Learner.

Super classes

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

Active bindings

internal_valid_scores The last observation of the validation scores for all metrics. Extracted from model$evaluation_log

internal_tuned_values Returns the early stopped iterations if early_stopping_rounds was set during training.

validate How to construct the internal validation data. This parameter can be either NULL, a ratio, "test", or "predefined".

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

```
# Define the Learner
learner = mlr3::lrn("regr.catboost")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.cforest

Regression Conditional Random Forest Learner

Description

A random forest based on conditional inference trees (ctree). Calls `partykit::cforest()` from **partykit**.

Dictionary

This Learner can be instantiated via lrn():

```
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 parameters

- `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

```
# Define the Learner
learner = mlr3::lrn("regr.cforest")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.ctree`

Regression Conditional Inference Tree Learner

Description

Regression Partition Tree where a significance test is used to determine the univariate splits. Calls `partykit::ctree()` from **partykit**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("regr.ctree")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.cubist`
Regression Cubist Learner

Description

Rule-based model that is an extension of Quinlan's M5 model tree. Each tree contains linear regression models at the terminal leaves. Calls `Cubist::cubist()` from **Cubist**.

Dictionary

This Learner can be instantiated via lrn():

```
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	-		$[1, 100]$
unbiased	logical	FALSE	TRUE, FALSE	-
rules	integer	100		$[1, \infty)$
extrapolation	numeric	100		$[0, 100]$
sample	integer	0		$[0, \infty)$
seed	integer	-		$(-\infty, \infty)$
label	untyped	"outcome"		-
neighbors	integer	-		$[0, 9]$

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

```
# Define the Learner
learner = mlr3::lrn("regr.cubist")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")
```

```
# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.decision_stump`
Regression Decision Stump Learner

Description

Decision Stump Learner. Calls `RWeka::DecisionStump()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.decision_stump")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrDecisionStump`

Methods

Public methods:

- `LearnerRegrDecisionStump$new()`
- `LearnerRegrDecisionStump$clone()`

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

Usage:

```
LearnerRegrDecisionStump$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.decision_stump")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.decision_table`

Regression Decision Table Learner

Description

Simple Decision Table majority regressor. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Initial parameter values

- `E`:
  - Has only 2 out of 4 original evaluation measures : rmse and mae with rmse being the default
  - Reason for change: this learner should only contain evaluation measures appropriate for regression tasks

Custom mlr3 parameters

- `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
- `P_best`:
  - original id: P
- `D_best`:
  - original id: D
- `N_best`:
  - original id: N
- `S_best`:
  - original id: S
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.decision_table")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
S	character	BestFirst	BestFirst, GreedyStepwise	-
X	integer	1		$(-\infty, \infty)$
E	character	rmse	rmse, mae	-
I	logical	-	TRUE, FALSE	-
R	logical	-	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		$[1, \infty)$
P_best	untyped	-		-
D_best	character	1	0, 1, 2	-
N_best	integer	-		$(-\infty, \infty)$
S_best	integer	1		$(-\infty, \infty)$
options	untyped	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrDecisionTable`

Methods

Public methods:

- `LearnerRegrDecisionTable$new()`
- `LearnerRegrDecisionTable$clone()`

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

Usage:

```
LearnerRegrDecisionTable$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Kohavi R (1995). "The Power of Decision Tables." In *8th European Conference on Machine Learning*, 174–189.

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

```
# Define the Learner
learner = mlr3::lrn("regr.decision_table")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.earth`

Implementation of Multivariate Adaptive Regression Splines

Description

This is an alternative implementation of MARS (Multivariate Adaptive Regression Splines). MARS is trademarked and thus not used as the name. The name "earth" stands for "Enhanced Adaptive Regression Through Hinges".

Calls `earth::earth()` from **earth**.

Details

Methods for variance estimations are not yet implemented.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.earth")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
wp	untyped	NULL		-
offset	untyped	NULL		-
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	NULL		-
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

```
# Define the Learner
learner = mlr3::lrn("regr.earth")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.fnn *Regression Fast Nearest Neighbor Search Learner*

Description

Fast Nearest Neighbour Regression. Calls `FNN::knn.reg()` from **FNN**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("regr.fnn")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.gam *Regression Generalized Additive Model Learner*

Description

Generalized additive models. Calls `mgcv::gam()` from package **mgcv**.

Formula

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.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.gam")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
family	character	gaussian	gaussian, poisson	-
formula	untyped	-		-
offset	untyped	NULL		-
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	NULL		-
sp	untyped	NULL		-
min.sp	untyped	NULL		-
H	untyped	NULL		-
gamma	numeric	1		$[1, \infty)$
paraPen	untyped	NULL		-
G	untyped	NULL		-
in.out	untyped	NULL		-
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	-
nei	untyped	-		-
ncv.threads	integer	1		$[1, \infty)$
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

Fit a generalized additive regression model using a boosting algorithm. Calls `mboost::gamboost()` from **mboost**.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-
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

```
learner = lrn("regr.gamboost", baselearner = "bols")
learner
```

`mlr_learners_regr.gaussian_processes`
Regression Gaussian Processes Learner From Weka

Description

Gaussian Processes. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `E_poly`:
  - original id: E
- `L_poly`:
  - original id: L (duplicated L for when K is set to PolyKernel)
- `C_poly`:
  - original id: C
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern
- `output-debug-info` for kernel parameter removed:
  - enables debugging output (if available) to be printed
- Reason for change: The parameter is removed because it's unclear how to actually use it.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.gaussian_processes")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
subset	untyped	-	
na.action	untyped	-	
L	numeric	1	
N	character	0	0, 1, 2
K	character	supportVector.PolyKernel	supportVector.NormalizedPolyKernel, supportVector.Pc
S	integer	1	
E_poly	numeric	1	
L_poly	logical	FALSE	TRUE, FALSE
C_poly	integer	250007	
output_debug_info	logical	FALSE	TRUE, FALSE
do_not_check_capabilities	logical	FALSE	TRUE, FALSE
num_decimal_places	integer	2	
batch_size	integer	100	
options	untyped	NULL	

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrGaussianProcesses`

Methods

Public methods:

- `LearnerRegrGaussianProcesses$new()`
- `LearnerRegrGaussianProcesses$clone()`

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

Usage:

```
LearnerRegrGaussianProcesses$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Mackay DJ (1998). "Introduction to Gaussian Processes."

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

```
# Define the Learner
learner = mlr3::lrn("regr.gaussian_processes")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.gausspr`

Regression Gaussian Process Learner

Description

Gaussian process for regression. Calls `kernlab::gausspr()` from **kernlab**. 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 lrn():

```
lrn("regr.gausspr")
```

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
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	
variance.model	logical	FALSE	TRUE, FALSE
tol	numeric	0.001	
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

```
# Define the Learner
learner = mlr3::lrn("regr.gausspr")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

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

Description

Gradient Boosting Regression Algorithm. Calls `gbm::gbm()` from **gbm**.

Dictionary

This Learner can be instantiated via lrn():

```
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	-		-

Parameter changes

- `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

```
# Define the Learner
learner = lrn("regr.gbm")
print(learner)
```

`mlr_learners_regr.glm` *Generalized Linear Regression*

Description

Generalized linear model. Calls `stats::glm()` from base package 'stats'. For logistic regression please use `mlr_learners_classif.log_reg`.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-
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	NULL		-
type	character	link	response, link, terms	-

Initial parameter values

- `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.

Examples

```
# Define the Learner
learner = mlr3::lrn("regr.glm")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.glmboost`

Boosted Generalized Linear Regression Learner

Description

Fit a generalized linear regression model using a boosting algorithm. Calls `mboost::glmboost()` from **mboost**.

Dictionary

This Learner can be instantiated via lrn():

```
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	NULL		-
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

```
# Define the Learner
learner = mlr3::lrn("regr.glmboost")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
```

```
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.IBk` *Regression IBk Learner*

Description

Instance based algorithm: K-nearest neighbours regression. Calls `RWeka::IBk()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.IBk")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
subset	untyped	-	
na.action	untyped	-	
weight	character	-	I, F
K	integer	1	
E	logical	FALSE	TRUE, FALSE
W	integer	0	
X	logical	FALSE	TRUE, FALSE
A	character	LinearNNSearch	BallTree, CoverTree, FilteredNeighbourSearch, KDTree, Linearl
output_debug_info	logical	FALSE	TRUE, FALSE

do_not_check_capabilities	logical	FALSE	TRUE, FALSE
num_decimal_places	integer	2	
batch_size	integer	100	
options	untyped	NULL	

Custom mlr3 parameters

- `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
- `weight`:
  - original id: I and F
- Reason for change: original `I` and `F` params are interdependent (`I` can only be `TRUE` when `F` is `FALSE` and vice versa). The easiest way to encode this is to combine `I` and `F` into one factor param.

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

```
# Define the Learner
learner = mlr3::lrn("regr.IBk")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.kstar` *Regression KStar Learner*

Description

Instance-based regressor which differs from other instance-based learners in that it uses an entropy-based distance function. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.kstar")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
B	integer	20		$(-\infty, \infty)$
E	logical	-	TRUE, FALSE	-
M	character	a	a, d, m, n	-

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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrKStar`

Methods

Public methods:

- `LearnerRegrKStar$new()`
- `LearnerRegrKStar$clone()`

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

Usage:

```
LearnerRegrKStar$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Cleary JG, Trigg LE (1995). "K*: An Instance-based Learner Using an Entropic Distance Measure." In *12th International Conference on Machine Learning*, 108-114.

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

```
# Define the Learner
learner = mlr3::lrn("regr.kstar")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.ksvm`

Regression Kernlab Support Vector Machine

Description

Support Vector Regression. Calls `kernlab::ksvm()` from **kernlab**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("regr.ksvm")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.liblinear`

L2-Regularized Support Vector Regression Learner

Description

L2 regularized support vector regression. Calls `LiblineaR::LiblineaR()` from **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 lrn():

```
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	-

Initial parameter values

- `svr_eps`:
  - Actual default: `NULL`
  - Initial value: 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

```
# Define the Learner
learner = mlr3::lrn("regr.liblinear")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.lightgbm`

Regression LightGBM Learner

Description

Gradient boosting algorithm. Calls `lightgbm::lightgbm()` from **lightgbm**. The list of parameters can be found here and in the documentation of `lightgbm::lgb.train()`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.lightgbm")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
objective	character	regression	regression, regression_l1, huber, fair, poisson, quantile, mape, gam
eval	untyped	-	
verbose	integer	1	
record	logical	TRUE	TRUE, FALSE
eval_freq	integer	1	
callbacks	untyped	-	
reset_data	logical	FALSE	TRUE, FALSE
boosting	character	gbdt	gbdt, rf, dart, goss
linear_tree	logical	FALSE	TRUE, FALSE
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
data_sample_strategy	character	bagging	bagging, goss
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	
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	NULL	
monotone_constraints_method	character	basic	basic, intermediate, advanced
monotone_penalty	numeric	0	
feature_contri	untyped	NULL	
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	-	
use_quantized_grad	logical	TRUE	TRUE, FALSE
num_grad_quant_bins	integer	4	
quant_train_renew_leaf	logical	FALSE	TRUE, FALSE
stochastic_rounding	logical	TRUE	TRUE, FALSE
serializable	logical	TRUE	TRUE, FALSE
max_bin	integer	255	
max_bin_by_feature	untyped	NULL	
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
forcedbins_filename	untyped	""	
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	
num_machines	integer	1	
local_listen_port	integer	12400	
time_out	integer	120	
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_predict	integer	0	
num_iteration_predict	integer	-1	
pred_early_stop	logical	FALSE	TRUE, FALSE
pred_early_stop_freq	integer	10	
pred_early_stop_margin	numeric	10	
num_iterations	integer	100	
early_stopping_rounds	integer	-	
early_stopping_min_delta	numeric	-	
first_metric_only	logical	FALSE	TRUE, FALSE

Initial parameter values

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

Early Stopping and Validation

Early stopping can be used to find the optimal number of boosting rounds. Set `early_stopping_rounds` to an integer value to monitor the performance of the model on the validation set while training. For information on how to configure the validation set, see the *Validation* section of `mlr3::Learner`. The internal validation measure can be set the `eval` parameter which should be a list of mlr3::Measures, functions, or strings for the internal lightgbm measures. If `first_metric_only` = `FALSE` (default), the learner stops when any metric fails to improve.

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrLightGBM`

Active bindings

`internal_valid_scores` The last observation of the validation scores for all metrics. Extracted from `model$evaluation_log`

`internal_tuned_values` Returns the early stopped iterations if `early_stopping_rounds` was set during training.

`validate` How to construct the internal validation data. This parameter can be either `NULL`, a ratio, `"test"`, or `"predefined"`.

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

```
# Define the Learner
learner = mlr3::lrn("regr.lightgbm")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.linear_regression

Regression Linear Regression Learner From Weka

Description

Linear Regression learner that uses the Akaike criterion for model selection and is able to deal with weighted instances. Calls `RWeka::LinearRegression()` **RWeka**.

Custom mlr3 parameters

- `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
- `additional_stats`:
  - original id: additional-stats

- use_qr:
  - original id: use-qr
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.linear_regression")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
S	character	0	0, 1, 2	-
C	logical	FALSE	TRUE, FALSE	-
R	numeric	1e-08		$(-\infty, \infty)$
minimal	logical	FALSE	TRUE, FALSE	-
additional_stats	logical	FALSE	TRUE, FALSE	-
use_qr	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	NULL		-

Super classes

mlr3::Learner -> mlr3::LearnerRegr -> LearnerRegrLinearRegression

Methods

Public methods:

- `LearnerRegrLinearRegression$new()`
- `LearnerRegrLinearRegression$clone()`

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

Usage:

```
LearnerRegrLinearRegression$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.linear_regression")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.lmer`

Regression Linear Mixed Effects Learner

Description

Linear model with random effects. Calls `lme4::lmer()` from **lme4**.

Formula

Although most mlr3 learners don't allow to specify the formula manually, and automatically set it by valling `task$formula()`, this learner allows to set the `formula` because it's core functionality depends it. This means that it might not always use all features that are available in the task. Be aware, that this can sometimes lead to unexpected error messages, because mlr3 checks the compatibility between the learner and the task on **all** available features.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.lmer")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "logical", "integer", "numeric", "factor"
- Required Packages: **mlr3**, **lme4**

Parameters

Id	Type	Default	Levels
formula	untyped	-	
REML	logical	TRUE	TRUE, FAL
start	untyped	NULL	
verbose	integer	0	

offset	untyped	NULL	
contrasts	untyped	NULL	
optimizer	character	nloptwrap	Nelder_Me
restart_edge	logical	FALSE	TRUE, FAL
boundary.tol	numeric	1e-05	
calc.derivs	logical	TRUE	TRUE, FAL
check.nobs.vs.rankZ	character	ignore	ignore, war
check.nobs.vs.nlev	character	stop	ignore, war
check.nlev.gtreq.5	character	ignore	ignore, war
check.nlev.gtr.1	character	stop	ignore, war
check.nobs.vs.nRE	character	stop	ignore, war
check.rankX	character	message+drop.cols	message+d
check.scaleX	character	warning	warning, st
check.formula.LHS	character	stop	ignore, war
check.conv.grad	untyped	"lme4::.makeCC(\"warning\", tol = 2e-3, relTol = NULL)"	
check.conv.singular	untyped	"lme4::.makeCC(action = \"message\", tol = formals(lme4::isSingular)$tol)"	
check.conv.hess	untyped	"lme4::.makeCC(action = \"warning\", tol = 1e-6)"	
optCtrl	untyped	list()	
newparams	untyped	NULL	
re.form	untyped	NULL	
random.only	logical	FALSE	TRUE, FAL
allow.new.levels	logical	FALSE	TRUE, FAL
na.action	untyped	"stats::na.pass"	

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrLmer`

Methods

Public methods:

- `LearnerRegrLmer$new()`
- `LearnerRegrLmer$clone()`

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

Usage:

```
LearnerRegrLmer$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

s-kganz

References

Bates, M D (2010). "lme4: Mixed-effects modeling with R."

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

```
# Define the Learner and set parameter values
learner = lrn("regr.lmer", formula = cmedv ~ (1 | town))

# Define a Task
task = tsk("boston_housing")

learner$train(task)
print(learner$model)
```

`mlr_learners_regr.m5p` *Regression M5P Learner*

Description

Implements base routines for generating M5 Model trees and rules. Calls `RWeka::M5P()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.m5p")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
N	logical	-	TRUE, FALSE	-
U	logical	-	TRUE, FALSE	-
R	logical	-	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		$[1, \infty)$
L	logical	-	TRUE, FALSE	-
options	untyped	NULL		-

Super classes

mlr3::Learner -> mlr3::LearnerRegr -> LearnerRegrM5P

Methods

Public methods:

- `LearnerRegrM5P$new()`
- `LearnerRegrM5P$clone()`

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

Usage:

```
LearnerRegrM5P$new()
```

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

Usage:

```
LearnerRegrM5P$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Quinlan RJ (1992). "Learning with Continuous Classes." In *5th Australian Joint Conference on Artificial Intelligence*, 343-348.

Wang Y, Witten IH (1997). "Induction of model trees for predicting continuous classes." In *Poster papers of the 9th European Conference on Machine Learning*.

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

```
# Define the Learner
learner = mlr3::lrn("regr.m5p")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")
```

```
# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.M5Rules`

Regression M5Rules Learner

Description

Algorithm for inducing decision lists from model trees. Calls `RWeka::M5Rules()` from **RWeka**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.M5Rules")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "integer", "numeric", "factor", "ordered"
- 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	NULL		-

Custom mlr3 parameters

- `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

```
# Define the Learner
learner = mlr3::lrn("regr.M5Rules")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.mars`

Regression Mars Learner

Description

Multivariate Adaptive Regression Splines. Calls `mda::mars()` from **mda**.

Dictionary

This Learner can be instantiated via lrn():

```
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

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

```
# Define the Learner
learner = mlr3::lrn("regr.mars")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.mob` *Regression Model-based Recursive Partitioning Learner*

Description

Model-based recursive partitioning algorithm. Calls `partykit::mob()` from **mob**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
library(mlr3)
lm_ = function(y, x, start = NULL, weights = NULL, offset = NULL, ...) {
  lm(y ~ 1, ...)
}
learner = LearnerRegrMob$new()
learner$param_set$values$rhs = "."
learner$param_set$values$fit = lm_
learner$feature_types = c("logical", "integer", "numeric", "factor", "ordered")

predict_fun = function(object, newdata, task, .type) {
  preds = predict(object, newdata = newdata, type = "response", se.fit = TRUE)
  cbind(preds$fit, preds$se.fit)
}
learner$param_set$values$predict_fun = predict_fun
task = tsk("mtcars")
ids = partition(task)
learner$train(task, row_ids = ids$train)
learner$predict(task, row_ids = ids$test)
```

---

mlr_learners_regr.multilayer_perceptron

Regression MultilayerPerceptron Learner

Description

Regressor that uses backpropagation to learn a multi-layer perceptron. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `G` removed:
  - GUI will be opened
- Reason for change: The parameter is removed because we don't want to launch GUI.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.multilayer_perceptron")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
L	numeric	0.3		$[0, 1]$
M	numeric	0.2		$[0, 1]$
N	integer	500		$[1, \infty)$
V	numeric	0		$[0, 100]$
S	integer	0		$[0, \infty)$
E	integer	20		$[1, \infty)$
A	logical	FALSE	TRUE, FALSE	-
B	logical	FALSE	TRUE, FALSE	-
H	untyped	"a"		-
C	logical	FALSE	TRUE, FALSE	-
I	logical	FALSE	TRUE, FALSE	-
R	logical	FALSE	TRUE, FALSE	-
D	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	NULL	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrMultilayerPerceptron`

Methods

Public methods:

- `LearnerRegrMultilayerPerceptron$new()`
- `LearnerRegrMultilayerPerceptron$clone()`

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

Usage:

```
LearnerRegrMultilayerPerceptron$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.multilayer_perceptron")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.priority_lasso`

Regression Priority Lasso Learner

Description

Patient outcome prediction based on multi-omics data taking practitioners' preferences into account. Calls `prioritylasso::prioritylasso()` from **prioritylasso**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.priority_lasso")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
blocks	untyped	-		-
max.coef	untyped	NULL		-
block1.penalization	logical	TRUE	TRUE, FALSE	-
lambda.type	character	lambda.min	lambda.min, lambda.1se	-
standardize	logical	TRUE	TRUE, FALSE	-
nfolds	integer	5		$[1, \infty)$
foldid	untyped	NULL		-
cvoffset	logical	FALSE	TRUE, FALSE	-
cvoffsetnfolds	integer	10		$[1, \infty)$
handle.missingtestdata	character	-	none, omit.prediction, set.zero, impute.block	-
include.allintercepts	logical	FALSE	TRUE, FALSE	-
use.blocks	untyped	"all"		-
alignment	character	lambda	lambda, fraction	-
alpha	numeric	1		$[0, 1]$
big	numeric	9.9e+35		$(-\infty, \infty)$
devmax	numeric	0.999		$[0, 1]$
dfmax	integer	-		$[0, \infty)$
eps	numeric	1e-06		$[0, 1]$
epsnr	numeric	1e-08		$[0, 1]$
exclude	untyped	-		-
exmx	numeric	250		$(-\infty, \infty)$
fdev	numeric	1e-05		$[0, 1]$
gamma	untyped	-		-
grouped	logical	TRUE	TRUE, FALSE	-
intercept	logical	TRUE	TRUE, FALSE	-
keep	logical	FALSE	TRUE, FALSE	-
lambda	untyped	-		-
lambda.min.ratio	numeric	-		$[0, 1]$
lower.limits	untyped	-Inf		-
maxit	integer	100000		$[1, \infty)$
mnlam	integer	5		$[1, \infty)$
mxit	integer	100		$[1, \infty)$
mxitnr	integer	25		$[1, \infty)$
nlambda	integer	100		$[1, \infty)$
offset	untyped	NULL		-
parallel	logical	FALSE	TRUE, FALSE	-
penalty.factor	untyped	-		-
pmax	integer	-		$[0, \infty)$
pmin	numeric	1e-09		$[0, 1]$
prec	numeric	1e-10		$(-\infty, \infty)$
standardize.response	logical	FALSE	TRUE, FALSE	-
thresh	numeric	1e-07		$[0, \infty)$
trace.it	integer	0		$[0, 1]$
type.gaussian	character	-	covariance, naive	-
type.logistic	character	Newton	Newton, modified.Newton	-

type.multinomial	character	ungrouped	ungrouped, grouped	-
upper.limits	untyped	Inf		-
scale.y	logical	FALSE	TRUE, FALSE	-
return.x	logical	TRUE	TRUE, FALSE	-
predict.gamma	numeric	gamma.1se		$(-\infty, \infty)$
relax	logical	FALSE	TRUE, FALSE	-
s	numeric	lambda.1se		$[0, 1]$

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrPriorityLasso`

Methods

Public methods:

- `LearnerRegrPriorityLasso$new()`
- `LearnerRegrPriorityLasso$selected_features()`
- `LearnerRegrPriorityLasso$clone()`

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

Usage:

```
LearnerRegrPriorityLasso$new()
```

Method `selected_features()`**:** Selected features when coef is positive

Usage:

```
LearnerRegrPriorityLasso$selected_features()
```

Returns: `character()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

HarutyunyanLiana

References

Simon K, Vindi J, Roman H, Tobias H, Anne-Laure B (2018). "Priority-Lasso: a simple hierarchical approach to the prediction of clinical outcome using multi-omics data." *BMC Bioinformatics*, **19**. doi:10.1186/s1285901823446.

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

```
# Define the Learner and set parameter values
learner = lrn("regr.priority_lasso",
  blocks = list(bp1 = 1:4, bp2 = 5:9, bp3 = 10:28, bp4 = 29:1028))
print(learner)

# Define a Task
task = mlr3::as_task_regr(prioritylasso::pl_data, target = "pl_out")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

# print the model
print(learner$model)

 # Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.randomForest

Regression Random Forest Learner

Description

Random forest for regression. Calls `randomForest::randomForest()` from **randomForest**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.randomForest")
```

Meta Information

- Task type: "regr"
- Predict Types: "response"
- Feature Types: "logical", "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	-
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

```
# Define the Learner
learner = mlr3::lrn("regr.randomForest")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.random_forest_weka

Regression Random Forest Learner from Weka

Description

Class for constructing a forest of random trees. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `store_out_of_bag_predictions`:
  - original id: store-out-of-bag-predictions

- `output_out_of_bag_complexity_statistics`:
  - original id: output-out-of-bag-complexity-statistics
- `num_slots`:
  - original id: num-slots
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern
- `attribute-importance` removed:
  - Compute and output attribute importance (mean impurity decrease method)
- Reason for change: The parameter is removed because it's unclear how to actually use it.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.random_forest_weka")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
P	numeric	100		$[0, 100]$
O	logical	FALSE	TRUE, FALSE	-
store_out_of_bag_predictions	logical	FALSE	TRUE, FALSE	-
output_out_of_bag_complexity_statistics	logical	FALSE	TRUE, FALSE	-
print	logical	FALSE	TRUE, FALSE	-
I	integer	100		$[1, \infty)$
num_slots	integer	1		$(-\infty, \infty)$
K	integer	0		$(-\infty, \infty)$
M	integer	1		$[1, \infty)$
V	numeric	0.001		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$
depth	integer	0		$[0, \infty)$
N	integer	0		$(-\infty, \infty)$
U	logical	FALSE	TRUE, FALSE	-
B	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	NULL	-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrRandomForestWeka`

Methods

Public methods:

- `LearnerRegrRandomForestWeka$new()`
- `LearnerRegrRandomForestWeka$clone()`

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

Usage:

```
LearnerRegrRandomForestWeka$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.random_forest_weka")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.random_tree`

Regression Random Tree Learner

Description

Tree that considers K randomly chosen attributes at each node. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.random_tree")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
K	integer	0		$[0, \infty)$
M	integer	1		$[1, \infty)$
V	numeric	0.001		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$
depth	integer	0		$[0, \infty)$
N	integer	0		$[0, \infty)$
U	logical	FALSE	TRUE, FALSE	-
B	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrRandomTree`

Methods

Public methods:

- `LearnerRegrRandomTree$new()`
- `LearnerRegrRandomTree$clone()`

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

Usage:

```
LearnerRegrRandomTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.random_tree")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.reptree`

Regression Decision Tree Learner

Description

Fast decision tree learner. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.reptree")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
M	integer	2		$(-\infty, \infty)$
V	numeric	0.001		$(-\infty, \infty)$
N	integer	3		$(-\infty, \infty)$
S	integer	1		$(-\infty, \infty)$

P	logical	-	TRUE, FALSE	-
L	integer	-1		$(-\infty, \infty)$
I	integer	0		$(-\infty, \infty)$
R	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrREPTree`

Methods

Public methods:

- `LearnerRegrREPTree$new()`
- `LearnerRegrREPTree$clone()`

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

Usage:

```
LearnerRegrREPTree$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.reptree")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.rfsrc`

Regression Random Forest SRC Learner

Description

Random forest for regression. Calls `randomForestSRC::rfsrc()` from **randomForestSRC**.

Dictionary

This Learner can be instantiated via lrn():

```
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		$[1, \infty)$
save.memory	logical	FALSE	TRUE, FALSE	-
perf.type	character	-	none	-
case.depth	logical	FALSE	TRUE, FALSE	-

Custom mlr3 parameters

- `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.
- `cores`: This value is set as the option `rf.cores` during training and is set to 1 by default.

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

```
# Define the Learner
learner = mlr3::lrn("regr.rfsrc")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.rpf` *Regression Random Planted Forest Learner*

Description

Random Planted Forest: A directly interpretable tree ensemble.

Calls `randomPlantedForest::rpf()` from 'randomPlantedForest'.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.rpf")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
max_interaction	integer	1		$[0, \infty)$
max_interaction_ratio	numeric	-		$[0, 1]$
max_interaction_limit	integer	-		$[1, \infty)$
ntrees	integer	50		$[1, \infty)$
splits	integer	30		$[1, \infty)$
split_try	integer	10		$[1, \infty)$
t_try	numeric	0.4		$[0, 1]$
deterministic	logical	FALSE	TRUE, FALSE	-
nthreads	integer	1		$[1, \infty)$
cv	logical	FALSE	TRUE, FALSE	-
purify	logical	FALSE	TRUE, FALSE	-

Custom mlr3 parameters

- `max_interaction`:

- This hyperparameter can alternatively be set via `max_interaction_ratio` as `max_interaction = max(ceiling(max_interaction_ratio * n_features), 1)`. The parameter `max_interaction_limit` can optionally be set as an upper bound, such that `max_interaction_ratio * min(n_features, max_interaction_limit)` is used instead. This is analogous to `mtry.ratio` in `classif.ranger`, with `max_interaction_limit` as an additional constraint. The parameter `max_interaction_limit` is initialized to `Inf`.

Installation

Package 'randomPlantedForest' is not on CRAN and has to be installed from GitHub via `remotes::install_github("Plan`

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrRandomPlantedForest`

Methods

Public methods:

- `LearnerRegrRandomPlantedForest$new()`
- `LearnerRegrRandomPlantedForest$clone()`

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

Usage:

```
LearnerRegrRandomPlantedForest$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

jemus42

References

Hiabu, Munir, Mammen, Enno, Meyer, T. J (2023). "Random Planted Forest: a directly interpretable tree ensemble." *arXiv preprint arXiv:2012.14563*. doi:10.48550/ARXIV.2012.14563.

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

```
# Define the Learner
learner = mlr3::lrn("regr.rpf")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.rsm` *Regression Response Surface Model Learner*

Description

Fit a linear model with a response-surface component. Calls `rsm::rsm()` from **rsm**.

Custom mlr3 parameters

- `modelfun`: This parameter controls how the formula for `rsm::rsm()` is created. Possible values are:
  - `"FO"` - first order
  - `"TWI"` - wo-way interactions, this is with 1st oder terms
  - `"SO"` - full second order

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.rsm")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
modelfun	character	-	FO, TWI, SO

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrRSM`

Methods

Public methods:

- `LearnerRegrRSM$new()`
- `LearnerRegrRSM$clone()`

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

Usage:

```
LearnerRegrRSM$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

sebffischer

References

Lenth, V R (2010). "Response-surface methods in R, using rsm." *Journal of Statistical Software*, **32**, 1–17.

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

```
# Define the Learner
learner = mlr3::lrn("regr.rsm")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.rvm` *Regression Relevance Vector Machine Learner*

Description

Bayesian version of the support vector machine. 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`. Calls `kernlab::rvm()` from package **kernlab**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.rvm")
```

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
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

```
# Define the Learner
learner = mlr3::lrn("regr.rvm")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.sgd` *Regression Stochastic Gradient Descent Learner*

Description

Stochastic Gradient Descent for learning various linear models. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Initial parameter values

- `F`:
  - Has only 3 out of 5 original loss functions: 2 = squared loss (regression), 3 = epsilon insensitive loss (regression) and 4 = Huber loss (regression) with 2 (squared loss) being the new default
  - Reason for change: this learner should only contain loss functions appropriate for regression tasks

Custom mlr3 parameters

- `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

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.sgd")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
F	character	2	2, 3, 4	-
L	numeric	0.01		$(-\infty, \infty)$
R	numeric	1e-04		$(-\infty, \infty)$
E	integer	500		$(-\infty, \infty)$
C	numeric	0.001		$(-\infty, \infty)$
N	logical	-	TRUE, FALSE	-
M	logical	-	TRUE, FALSE	-
S	integer	1		$(-\infty, \infty)$
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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrSGD`

Methods

Public methods:

- `LearnerRegrSGD$new()`
- `LearnerRegrSGD$clone()`

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

Usage:

```
LearnerRegrSGD$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.sgd")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_regr.simple_linear_regression

Regression SimpleLinearRegression Learner from Weka

Description

Simple linear regression model that picks the attribute that results in the lowest squared error. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `additional_stats`:
  - original id: additional-stats
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.simple_linear_regression")
```

Meta Information

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

Parameters

Id	Type	Default	Levels	Range
subset	untyped	-		-
na.action	untyped	-		-
additional_stats	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	NULL		-

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrSimpleLinearRegression`

Methods

Public methods:

- `LearnerRegrSimpleLinearRegression$new()`
- `LearnerRegrSimpleLinearRegression$clone()`

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

Usage:

```
LearnerRegrSimpleLinearRegression$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

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

```
# Define the Learner
learner = mlr3::lrn("regr.simple_linear_regression")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_regr.smo_reg`
Regression Support Vector Machine Learner

Description

Support Vector Machine for regression. Calls `RWeka::make_Weka_classifier()` from **RWeka**.

Custom mlr3 parameters

- `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
- `T_improved`:
  - original id: T
- `V_improved`:
  - original id: V
- `P_improved`:
  - original id: P
- `L_improved`:
  - original id: L (duplicated L for when I is set to RegSMOImproved)
- `W_improved`:
  - original id: W
- `C_poly`:
  - original id: C
- `E_poly`:

  - original id: E
- `L_poly`:
  - original id: L (duplicated L for when K is set to PolyKernel)
- Reason for change: This learner contains changed ids of the following control arguments since their ids contain irregular pattern

Dictionary

This Learner can be instantiated via lrn():

```
lrn("regr.smo_reg")
```

Meta Information

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

Parameters

Id	Type	Default	Levels
subset	untyped	-	
na.action	untyped	-	
C	numeric	1	
N	character	0	0, 1, 2
I	character	RegSMOImproved	RegSMO, RegSMOImproved
K	character	PolyKernel	NormalizedPolyKernel, PolyKernel, Puk, RBFKernel, StringK
T_improved	numeric	0.001	
V_improved	logical	TRUE	TRUE, FALSE
P_improved	numeric	1e-12	
L_improved	numeric	0.001	
W_improved	integer	1	
C_poly	integer	250007	
E_poly	numeric	1	
L_poly	logical	FALSE	TRUE, FALSE
output_debug_info	logical	FALSE	TRUE, FALSE
do_not_check_capabilities	logical	FALSE	TRUE, FALSE
num_decimal_places	integer	2	
batch_size	integer	100	
options	untyped	NULL	

Super classes

`mlr3::Learner` -> `mlr3::LearnerRegr` -> `LearnerRegrSMOreg`

Methods

Public methods:

- `LearnerRegrSMOreg$new()`
- `LearnerRegrSMOreg$clone()`

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

Usage:

```
LearnerRegrSMOreg$new()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

damirpolat

References

Shevade S, Keerthi S, Bhattacharyya C, Murthy K (1999). "Improvements to the SMO Algorithm for SVM Regression." In *IEEE Transactions on Neural Networks*.

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

```
# Define the Learner
learner = mlr3::lrn("regr.smo_reg")
print(learner)

# Define a Task
task = mlr3::tsk("mtcars")

# Create train and test set
```

```
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.akritas`
Survival Akritas Estimator Learner

Description

Survival akritas estimator. Calls `survivalmodels::akritas()` from package 'survivalmodels'.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.akritas")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "numeric", "character", "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	-
ntime	numeric	150		$[1, \infty)$
round_time	integer	2		$[0, \infty)$

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

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

```
# Define the Learner
learner = mlr3::lrn("surv.akritas")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.aorsf`

Accelerated Oblique Random Survival Forest Learner

Description

Accelerated oblique random survival forest. Calls `aorsf::orsf()` from **aorsf**. Note that although the learner has the property `"missing"` and it can in principle deal with missing values, the behaviour has to be configured using the parameter `na_action`.

Details

This learner returns three prediction types:

1. `distr`: a survival matrix in two dimensions, where observations are represented in rows and (unique event) time points in columns.
2. `response`: the restricted mean survival time of each test observation, derived from the survival matrix prediction (`distr`).
3. `crank`: the expected mortality using mlr3proba::.surv_return.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.aorsf")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”, “response”
- Feature Types: “integer”, “numeric”, “factor”, “ordered”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **aorsf**, **pracma**

Parameters

Id	Type	Default	Levels	Range
n_tree	integer	500		$[1, \infty)$
n_split	integer	5		$[1, \infty)$
n_retry	integer	3		$[0, \infty)$
n_thread	integer	0		$[0, \infty)$
pred_aggregate	logical	TRUE	TRUE, FALSE	-
pred_simplify	logical	FALSE	TRUE, FALSE	-
oobag	logical	FALSE	TRUE, FALSE	-
mtry	integer	NULL		$[1, \infty)$
mtry_ratio	numeric	-		$[0, 1]$
sample_with_replacement	logical	TRUE	TRUE, FALSE	-
sample_fraction	numeric	0.632		$[0, 1]$
control_type	character	fast	fast, cph, net	-
split_rule	character	logrank	logrank, cstat	-
control_fast_do_scale	logical	FALSE	TRUE, FALSE	-
control_fast_ties	character	efron	efron, breslow	-
control_cph_ties	character	efron	efron, breslow	-
control_cph_eps	numeric	1e-09		$[0, \infty)$
control_cph_iter_max	integer	20		$[1, \infty)$
control_net_alpha	numeric	0.5		$(-\infty, \infty)$
control_net_df_target	integer	NULL		$[1, \infty)$
leaf_min_events	integer	1		$[1, \infty)$
leaf_min_obs	integer	5		$[1, \infty)$
split_min_events	integer	5		$[1, \infty)$
split_min_obs	integer	10		$[1, \infty)$
split_min_stat	numeric	NULL		$[0, \infty)$
oobag_pred_type	character	risk	none, surv, risk, chf, mort	-
importance	character	anova	none, anova, negate, permute	-
importance_max_pvalue	numeric	0.01		$[1e-04, 0.9999]$
tree_seeds	integer	NULL		$[1, \infty)$
oobag_pred_horizon	numeric	NULL		$[0, \infty)$
oobag_eval_every	integer	NULL		$[1, \infty)$
oobag_fun	untyped	NULL		-
attach_data	logical	TRUE	TRUE, FALSE	-
verbose_progress	logical	FALSE	TRUE, FALSE	-
na_action	character	fail	fail, omit, impute_meanmode	-

Initial parameter values

- `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` -> `LearnerSurvAorsf`

Methods

Public methods:

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

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

Usage:

```
LearnerSurvAorsf$new()
```

Method `oob_error()`**:** OOB concordance error extracted from the model slot `eval_oobag$stat_values`

Usage:

```
LearnerSurvAorsf$oob_error()
```

Returns: `numeric()`.

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

Usage:

```
LearnerSurvAorsf$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

bcjaeger

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.

Jaeger BC, Welden S, Lenoir K, Speiser JL, Segar MW, Pandey A, Pajewski NM (2023). "Accelerated and interpretable oblique random survival forests." *Journal of Computational and Graphical Statistics*, 1–16. doi:10.1080/10618600.2023.2231048.

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

```
# Define the Learner
learner = mlr3::lrn("surv.aorsf")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.bart`

Survival Bayesian Additive Regression Trees Learner

Description

Fits a Bayesian Additive Regression Trees (BART) learner to right-censored survival data. Calls `BART::mc.surv.bart()` from **BART**.

Details

Two types of prediction are returned for this learner:

1. `distr`: a 3d survival array with observations as 1st dimension, time points as 2nd and the posterior draws as 3rd dimension.
2. `crank`: the expected mortality using mlr3proba::.surv_return. The parameter `which.curve` decides which posterior draw (3rd dimension) will be used for the calculation of the expected mortality. Note that the median posterior is by default used for the calculation of survival measures that require a `distr` prediction, see more info on PredictionSurv.

Initial parameter values

- `mc.cores` is initialized to 1 to avoid threading conflicts with **future**.

Custom mlr3 parameters

- `quiet` allows to suppress messages generated by the wrapped C++ code. Is initialized to `TRUE`.
- `importance` allows to choose the type of importance. Default is `count`, see documentation of method `$importance()` for more details.
- `which.curve` allows to choose which posterior draw will be used for the calculation of the `crank` prediction. If between (0,1) it is taken as the quantile of the curves otherwise if greater than 1 it is taken as the curve index, can also be 'mean'. By default the **median posterior** is used, i.e. `which.curve` is 0.5.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.bart")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **BART**

Parameters

Id	Type	Default	Levels	Range
K	numeric	NULL		$[1, \infty)$
events	untyped	NULL		-
ztimes	untyped	NULL		-
zdelta	untyped	NULL		-
sparse	logical	FALSE	TRUE, FALSE	-
theta	numeric	0		$(-\infty, \infty)$
omega	numeric	1		$(-\infty, \infty)$
a	numeric	0.5		$[0.5, 1]$
b	numeric	1		$(-\infty, \infty)$
augment	logical	FALSE	TRUE, FALSE	-
rho	numeric	NULL		$(-\infty, \infty)$
usequants	logical	FALSE	TRUE, FALSE	-
rm.const	logical	TRUE	TRUE, FALSE	-
type	character	pbart	pbart, lbart	-
ntype	integer	-		$[1, 3]$
k	numeric	2		$[0, \infty)$
power	numeric	2		$[0, \infty)$
base	numeric	0.95		$[0, 1]$
offset	numeric	NULL		$(-\infty, \infty)$
ntree	integer	50		$[1, \infty)$
numcut	integer	100		$[1, \infty)$
ndpost	integer	1000		$[1, \infty)$
nskip	integer	250		$[0, \infty)$
keepevery	integer	10		$[1, \infty)$
printevery	integer	100		$[1, \infty)$
seed	integer	99		$(-\infty, \infty)$
mc.cores	integer	2		$[1, \infty)$
nice	integer	19		$[0, 19]$
openmp	logical	TRUE	TRUE, FALSE	-
quiet	logical	TRUE	TRUE, FALSE	-
importance	character	count	count, prob	-
which.curve	numeric	-		$[0, \infty)$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvLearnerSurvBART`

Methods

Public methods:

- `LearnerSurvLearnerSurvBART$new()`
- `LearnerSurvLearnerSurvBART$importance()`

- `LearnerSurvLearnerSurvBART$clone()`

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

Usage:

```
LearnerSurvLearnerSurvBART$new()
```

Method `importance()`**:** Two types of importance scores are supported based on the value of the parameter `importance`:

1. `prob`: The mean selection probability of each feature in the trees, extracted from the slot `varprob.mean`. If `sparse = FALSE` (default), this is a fixed constant. Recommended to use this option when `sparse = TRUE`.
2. `count`: The mean observed count of each feature in the trees (average number of times the feature was used in a tree decision rule across all posterior draws), extracted from the slot `varcount.mean`. This is the default importance scores.

In both cases, higher values signify more important variables.

Usage:

```
LearnerSurvLearnerSurvBART$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

bblodfon

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.

Chipman, A H, George, I E, McCulloch, E R (2010). "BART: Bayesian additive regression trees." *The Annals of Applied Statistics*, **4**(1), 266–298.

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

```
lrn("surv.bart")
```

`mlr_learners_surv.blackboost`
Gradient Boosting with Regression Trees Survival Learner

Description

Gradient boosting with regression trees for survival analysis. Calls `mboost::blackboost()` from **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.blackboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "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

```
# Define the Learner
learner = mlr3::lrn("surv.blackboost")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.cforest`

Survival Conditional Random Forest Learner

Description

A random forest based on conditional inference trees (ctree). Calls `partykit::cforest()` from **partykit**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.cforest")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- 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 parameters

- `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

```
task = tsk("rats")
learner = lrn("surv.cforest", ntree = 50)
splits = partition(task)
learner$train(task, splits$train)
pred = learner$predict(task, splits$test)
```

`mlr_learners_surv.coxboost`
Survival Cox Model with Likelihood Based Boosting Learner

Description

Fit a Survival Cox model with a likelihood based boosting algorithm. 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.

Three prediction types are returned for this learner, using the internal `predict.CoxBoost()` function:

1. `lp`: a vector of linear predictors (relative risk scores), one per observation.
2. `crank`: same as `lp`.
3. `distr`: a 2d survival matrix, with observations as rows and time points as columns. The internal transformation uses the Breslow estimator to compose the survival distributions from the `lp` predictions.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.coxboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "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	-		-

Installation

The package 'CoxBoost' is not on CRAN and has to be installed from GitHub using `remotes::install_github("binderh/`

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCoxboost`

Methods

Public methods:

- `LearnerSurvCoxboost$new()`
- `LearnerSurvCoxboost$selected_features()`
- `LearnerSurvCoxboost$clone()`

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

Usage:

```
LearnerSurvCoxboost$new()
```

Method `selected_features()`**:** Returns the set of selected features which have non-zero coefficients. Calls the internal `coef.CoxBoost()` function.

Usage:

```
LearnerSurvCoxboost$selected_features(at_step = NULL)
```

Arguments:

`at_step (integer(1))`
Which boosting step to get the coefficients for. If no step is given (default), the final boosting step is used.

Returns: (`character()`) vector of feature names.

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

```
# Define the Learner
learner = mlr3::lrn("surv.coxboost")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.coxtime`
Survival Cox-Time Learner

Description

Cox-Time survival model. Calls `survivalmodels::coxtime()` from package 'survivalmodels'.

Dictionary

This Learner can be instantiated via lrn():

```
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(32L, 32L)	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leaky
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	

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

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

```
lrn("surv.coxtime")
```

---

mlr_learners_surv.ctree

Survival Conditional Inference Tree Learner

Description

Survival Partition Tree where a significance test is used to determine the univariate splits. Calls `partykit::ctree()` from **partykit**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.ctree")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- 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

```
# Define the Learner
learner = mlr3::lrn("surv.ctree")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.cv_coxboost`
Survival Cox Model with Cross-Validation Likelihood Based Boosting Learner

---

Description

Fits a survival Cox model using likelihood based boosting and interal cross-validation for the number of steps. Calls `CoxBoost::CoxBoost()` or `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.

Three prediction types are returned for this learner, using the internal `predict.CoxBoost()` function:

1. `lp`: a vector of linear predictors (relative risk scores), one per observation.
2. `crank`: same as `lp`.

3. `distr`: a 2d survival matrix, with observations as rows and time points as columns. The internal transformation uses the Breslow estimator to compose the survival distributions from the `lp` predictions.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.cv_coxboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "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	NULL		-
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	-		-

Installation

The package 'CoxBoost' is not on CRAN and has to be installed from GitHub using `remotes::install_github("binderh/`

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCVCoxboost`

Methods

Public methods:

- `LearnerSurvCVCoxboost$new()`
- `LearnerSurvCVCoxboost$selected_features()`
- `LearnerSurvCVCoxboost$clone()`

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

Usage:

```
LearnerSurvCVCoxboost$new()
```

Method `selected_features()`**:** Returns the set of selected features which have non-zero coefficients. Calls the internal `coef.CoxBoost()` function.

Usage:

```
LearnerSurvCVCoxboost$selected_features(at_step = NULL)
```

Arguments:

`at_step (integer(1))`
Which boosting step to get the coefficients for. If no step is given (default), the final boosting step is used.

Returns: (`character()`) vector of feature names.

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

```
task = tsk("rats")
task$col_roles$feature = c("litter", "rx")
learner = lrn("surv.cv_coxboost", maxstepno = 20)
splits = partition(task)
learner$train(task, splits$train)
pred = learner$predict(task, splits$test)
```

`mlr_learners_surv.cv_glmnet` *Cross-Validated GLM with Elastic Net Regularization Survival Learner*

Description

Generalized linear models with elastic net regularization. Calls `glmnet::cv.glmnet()` from package **glmnet**.

Details

This learner returns two prediction types:

1. `lp`: a vector of linear predictors (relative risk scores), one per observation. Calculated using `glmnet::predict.cv.glmnet()`.
2. `distr`: a survival matrix in two dimensions, where observations are represented in rows and time points in columns. Calculated using `glmnet::survfit.cv.glmnet()`. Parameters `stype` and `ctype` relate to how `lp` predictions are transformed into survival predictions and are described in `survival::survfit.coxph()`. By default the Breslow estimator is used.

Custom mlr3 parameters

- `family` is set to `"cox"` and cannot be changed.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.cv_glmnet")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”, “lp”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **glmnet**

Parameters

Id	Type	Default	Levels	Range
alignment	character	lambda	lambda, fraction	-
alpha	numeric	1		$[0, 1]$
big	numeric	9.9e+35		$(-\infty, \infty)$
devmax	numeric	0.999		$[0, 1]$
dfmax	integer	-		$[0, \infty)$
eps	numeric	1e-06		$[0, 1]$
epsnr	numeric	1e-08		$[0, 1]$
exclude	untyped	-		-
exmx	numeric	250		$(-\infty, \infty)$
fdev	numeric	1e-05		$[0, 1]$
foldid	untyped	NULL		-
gamma	untyped	-		-
grouped	logical	TRUE	TRUE, FALSE	-
intercept	logical	TRUE	TRUE, FALSE	-
keep	logical	FALSE	TRUE, FALSE	-
lambda	untyped	-		-
lambda.min.ratio	numeric	-		$[0, 1]$
lower.limits	untyped	-Inf		-
maxit	integer	100000		$[1, \infty)$
mnlam	integer	5		$[1, \infty)$
mxit	integer	100		$[1, \infty)$
mxitnr	integer	25		$[1, \infty)$
nfolds	integer	10		$[3, \infty)$
nlambda	integer	100		$[1, \infty)$
offset	untyped	NULL		-
newoffset	untyped	-		-
parallel	logical	FALSE	TRUE, FALSE	-
penalty.factor	untyped	-		-
pmax	integer	-		$[0, \infty)$
pmin	numeric	1e-09		$[0, 1]$
prec	numeric	1e-10		$(-\infty, \infty)$
predict.gamma	numeric	gamma.1se		$(-\infty, \infty)$
relax	logical	FALSE	TRUE, FALSE	-
s	numeric	lambda.1se		$[0, \infty)$
standardize	logical	TRUE	TRUE, FALSE	-
standardize.response	logical	FALSE	TRUE, FALSE	-
thresh	numeric	1e-07		$[0, \infty)$
trace.it	integer	0		$[0, 1]$

type.gaussian	character	-	covariance, naive	-
type.logistic	character	Newton	Newton, modified.Newton	-
type.measure	character	deviance	deviance, C	-
type.multinomial	character	ungrouped	ungrouped, grouped	-
upper.limits	untyped	Inf		-
stype	integer	2		$[1, 2]$
ctype	integer	-		$[1, 2]$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvCVGlmnet`

Methods

Public methods:

- `LearnerSurvCVGlmnet$new()`
- `LearnerSurvCVGlmnet$selected_features()`
- `LearnerSurvCVGlmnet$clone()`

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

Usage:

```
LearnerSurvCVGlmnet$new()
```

Method `selected_features()`**:** Returns the set of selected features as reported by `glmnet::predict.glmnet()` with `type` set to `"nonzero"`.

Usage:

```
LearnerSurvCVGlmnet$selected_features(lambda = NULL)
```

Arguments:

`lambda (numeric(1))`
Custom `lambda`, defaults to the active lambda depending on parameter set.

Returns: (`character()`) of feature names.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

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

```
# Define the Learner
learner = mlr3::lrn("surv.cv_glmnet")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.deephit`

Survival DeepHit Learner

Description

Neural network 'Deephit' for survival analysis. Calls `survivalmodels::deephit()` from pacakge '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 lrn():

```
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(32L, 32L)	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyre
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	

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

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

```
lrn("surv.deephit")
```

---

mlr_learners_surv.deepsurv

Survival DeepSurv Learner

Description

DeepSurv fits a neural network based on the partial likelihood from a Cox PH. Calls `survivalmodels::dnnsurv()` from package 'survivalmodels'.

Dictionary

This Learner can be instantiated via lrn():

```
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(32L, 32L)	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyre
device	untyped	-	
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	

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

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

```
lrn("surv.deepsurv")
```

`mlr_learners_surv.dnnsurv` *Survival DNNSurv Learner*

Description

Fits a neural network based on pseudo-conditional survival probabilities. 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 lrn():

```
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	Rang
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	-		$(-\infty$
clipvalue	numeric	-		$(-\infty$
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	-		$(-\infty$
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	-		-
rho	numeric	0.95		$(-\infty$
global_clipnorm	numeric	-		$(-\infty$
use_ema	logical	-	TRUE, FALSE	-
ema_momentum	numeric	0.99		$(-\infty$
ema_overwrite_frequency	numeric	-		$(-\infty$
jit_compile	logical	TRUE	TRUE, FALSE	-
initial_accumultator_value	numeric	0.1		$(-\infty$
amsgrad	logical	FALSE	TRUE, FALSE	-
lr_power	numeric	-0.5		$(-\infty$
l1_regularization_strength	numeric	0		$[0, \infty$
l2_regularization_strength	numeric	0		$[0, \infty$
l2_shrinkage_regularization_strength	numeric	0		$[0, \infty$
beta	numeric	0		$(-\infty$
centered	logical	FALSE	TRUE, FALSE	-

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

Initial parameter values

- `verbose` is initialized to 0.

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

```
lrn("surv.dnnsurv")
```

`mlr_learners_surv.flexible`

Survival Flexible Parametric Spline Learner

Description

Flexible parametric spline learner. Calls `flexsurv::flexsurvspline()` from **flexsurv**.

Details

This learner returns two prediction types:

1. `lp`: a vector of linear predictors (relative risk scores), for each test observation. Calculated using `flexsurv::flexsurvspline()` and the estimated coefficients. 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, the linear predictor (`lp`) is $lp = \beta X$.
2. `distr`: a survival matrix in two dimensions, where observations are represented in rows and time points in columns. Calculated using `predict.flexsurvreg()`.
3. `crank`: same as `lp`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.flexible")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "lp"
- Feature Types: "logical", "integer", "numeric", "factor"
- 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	-
spline	character	rp	rp, splines2ns	-
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	$(-\infty, \infty)$

Initial parameter values

- k:
  - Actual default: `0`
  - Initial value: `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

```
lrn("surv.flexible")
```

---

mlr_learners_surv.gamboost

Boosted Generalized Additive Survival Learner

Description

Fits a generalized additive survival model using a boosting algorithm. Calls `mboost::gamboost()` from **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.gamboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "lp"
- Feature Types: "logical", "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	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	NULL		-
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

```
lrn("surv.gamboost")
```

---

mlr_learners_surv.gbm *Survival Gradient Boosting Machine Learner*

Description

Gradient Boosting for Survival Analysis. Calls `gbm::gbm()` from **gbm**.

Dictionary

This Learner can be instantiated via lrn():

```
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	FALSE	TRUE, FALSE	-
verbose	logical	FALSE	TRUE, FALSE	-
var.monotone	untyped	-		-
n.cores	integer	1		$(-\infty, \infty)$
single.tree	logical	FALSE	TRUE, FALSE	-

Parameter changes

- `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

```
# Define the Learner
learner = mlr3::lrn("surv.gbm")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.glmboost`

Boosted Generalized Linear Survival Learner

Description

Fits a generalized linear survival model using a boosting algorithm. Calls `mboost::glmboost()` from **mboost**.

Details

This learner returns up to three prediction types:

1. `crank`: same as `lp`.
2. `lp`: a vector of linear predictors (relative risk scores), one per observation. Calculated using `mboost::predict.glmboost()`.
3. `distr`: a survival matrix in two dimensions, where rows are observations and columns are the time points. This predict type is returned only when the `family` parameter is set to `"coxph"` (which is the default). Calculated using `mboost::survFit()` which uses the Breslow estimator for the baseline hazard function.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.glmboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "lp"
- Feature Types: "logical", "integer", "numeric", "factor"
- 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	NULL		-
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$new()`
- `LearnerSurvGLMBoost$importance()`
- `LearnerSurvGLMBoost$selected_features()`
- `LearnerSurvGLMBoost$clone()`

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

Usage:

```
LearnerSurvGLMBoost$new()
```

Method `importance()`**:** Importance scores are extracted with the function `mboost::varimp()` and represent a feature's individual contribution to the risk reduction per boosting step of the fitted model. The higher the risk reduction, the larger the feature importance.

Note: Importance is supported only for datasets with `numeric` features, as the presence of factors with multiple levels makes it difficult to get the original feature names.

Usage:

```
LearnerSurvGLMBoost$importance()
```

Returns: Named `numeric()`.

Method `selected_features()`**:** Selected features are extracted with the function `mboost::coef.glmboost()` which by default returns features with non-zero coefficients and for the final number of boosting iterations.

Note: Selected features can be retrieved only for datasets with `numeric` features, as the presence of factors with multiple levels makes it difficult to get the original feature names.

Usage:

```
LearnerSurvGLMBoost$selected_features()
```

Returns: `character()`.

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

```
# Define the Learner
learner = mlr3::lrn("surv.glmboost")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.glmnet`

GLM with Elastic Net Regularization Survival Learner

Description

Generalized linear models with elastic net regularization. Calls `glmnet::glmnet()` from package **glmnet**.

Details

This learner returns two prediction types:

1. `lp`: a vector of linear predictors (relative risk scores), one per observation. Calculated using `glmnet::predict.coxnet()`.
2. `distr`: a survival matrix in two dimensions, where observations are represented in rows and time points in columns. Calculated using `glmnet::survfit.coxnet()`. Parameters `stype` and `ctype` relate to how `lp` predictions are transformed into survival predictions and are described in `survival::survfit.coxph()`. By default the Breslow estimator is used.

Caution: This learner is different to learners calling `glmnet::cv.glmnet()` in that it does not use the internal optimization of parameter `lambda`. Instead, `lambda` needs to be tuned by the user (e.g., via **mlr3tuning**). When `lambda` is tuned, the `glmnet` will be trained for each tuning iteration. While fitting the whole path of `lambda`s would be more efficient, as is done by default in

`glmnet::glmnet()`, tuning/selecting the parameter at prediction time (using parameter `s`) is currently not supported in **mlr3** (at least not in efficient manner). Tuning the `s` parameter is, therefore, currently discouraged.

When the data are i.i.d. and efficiency is key, we recommend using the respective auto-tuning counterpart in `mlr_learners_surv.cv_glmnet()`. However, in some situations this is not applicable, usually when data are imbalanced or not i.i.d. (longitudinal, time-series) and tuning requires custom resampling strategies (blocked design, stratification).

Custom mlr3 parameters

- `family` is set to `"cox"` and cannot be changed.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.glmnet")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”, “lp”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **glmnet**

Parameters

Id	Type	Default	Levels	Range
alignment	character	lambda	lambda, fraction	-
alpha	numeric	1		$[0, 1]$
big	numeric	9.9e+35		$(-\infty, \infty)$
devmax	numeric	0.999		$[0, 1]$
dfmax	integer	-		$[0, \infty)$
eps	numeric	1e-06		$[0, 1]$
epsnr	numeric	1e-08		$[0, 1]$
exact	logical	FALSE	TRUE, FALSE	-
exclude	untyped	-		-
exmx	numeric	250		$(-\infty, \infty)$
fdev	numeric	1e-05		$[0, 1]$
gamma	untyped	-		-
grouped	logical	TRUE	TRUE, FALSE	-
intercept	logical	TRUE	TRUE, FALSE	-
keep	logical	FALSE	TRUE, FALSE	-
lambda	untyped	-		-
lambda.min.ratio	numeric	-		$[0, 1]$
lower.limits	untyped	-Inf		-
maxit	integer	100000		$[1, \infty)$

mnlam	integer	5		$[1, \infty)$
mxit	integer	100		$[1, \infty)$
mxitnr	integer	25		$[1, \infty)$
newoffset	untyped	-		-
nlambda	integer	100		$[1, \infty)$
offset	untyped	NULL		-
parallel	logical	FALSE	TRUE, FALSE	-
penalty.factor	untyped	-		-
pmax	integer	-		$[0, \infty)$
pmin	numeric	1e-09		$[0, 1]$
prec	numeric	1e-10		$(-\infty, \infty)$
predict.gamma	numeric	gamma.1se		$(-\infty, \infty)$
relax	logical	FALSE	TRUE, FALSE	-
s	numeric	0.01		$[0, \infty)$
standardize	logical	TRUE	TRUE, FALSE	-
thresh	numeric	1e-07		$[0, \infty)$
trace.it	integer	0		$[0, 1]$
type.logistic	character	Newton	Newton, modified.Newton	-
type.multinomial	character	ungrouped	ungrouped, grouped	-
upper.limits	untyped	Inf		-
stype	integer	2		$[1, 2]$
ctype	integer	-		$[1, 2]$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvGlmnet`

Methods

Public methods:

- `LearnerSurvGlmnet$new()`
- `LearnerSurvGlmnet$selected_features()`
- `LearnerSurvGlmnet$clone()`

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

Usage:

```
LearnerSurvGlmnet$new()
```

Method `selected_features()`**:** Returns the set of selected features as reported by `glmnet::predict.glmnet()` with `type` set to `"nonzero"`.

Usage:

```
LearnerSurvGlmnet$selected_features(lambda = NULL)
```

Arguments:

`lambda (numeric(1))`
Custom `lambda`, defaults to the active lambda depending on parameter set.

Returns: (`character()`) of feature names.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

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

```
# Define the Learner
learner = mlr3::lrn("surv.glmnet")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
```

```
# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.loghaz`

Survival Logistic-Hazard Learner

Description

Survival logistic hazard learner. 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 lrn():

```
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(32L, 32L)	
batch_norm	logical	TRUE	TRUE, FALSE
dropout	numeric	-	
activation	character	relu	celu, elu, gelu, glu, hardshrink, hardsigmoid, hardswish, hardtanh, relu6, leakyre
custom_net	untyped	-	
device	untyped	-	
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	

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

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

```
lrn("surv.loghaz")
```

`mlr_learners_surv.mboost`

Boosted Generalized Additive Survival Learner

Description

Model-based boosting for survival analysis. Calls `mboost::mboost()` from **mboost**.

Details

`distr` prediction made by `mboost::survFit()`.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.mboost")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr", "lp"
- Feature Types: "logical", "integer", "numeric", "factor"
- 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	NULL		-
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

```
lrn("surv.mboost")
```

---

mlr_learners_surv.nelson

Survival Nelson-Aalen Estimator Learner

Description

Non-parametric estimator of the cumulative hazard rate function. Calls `survival::survfit()` from **survival**.

Dictionary

This Learner can be instantiated via lrn():

```
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

```
# Define the Learner
learner = mlr3::lrn("surv.nelson")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.parametric`

Survival Fully Parametric Learner

Description

Parametric survival model. Calls `parametric()]` from 'survivalmodels'.

Details

This learner allows you to choose a distribution and a model form to compose a predicted survival probability distribution.

The predict method is implemented in `survivalmodels::predict.parametric()`. Our implementation is more efficient for composition to distributions than `survival::predict.survreg()`.

Three types of prediction are returned for this learner:

1. `lp`: a vector of linear predictors (relative risk scores), one per test observation. `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.
2. `crank`: same as `lp`.
3. `distr`: a survival matrix in two dimensions, where observations are represented in rows and time points in columns. The distribution `distr` is composed using the `lp` predictions and specifying a model form in the `form` hyper-parameter. 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`)

$$S(t) = 1 - \Phi((t - lp)/s)$$

where S_0 is the estimated baseline survival distribution (in this case with a given parametric form), lp is the predicted linear predictor, Φ 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' (we note that ones mostly used in the literature):

- dist = "gaussian"; form = "tobit";
- dist = "weibull"; form = "ph"; (fairly used)
- dist = "exponential"; form = "ph";
- dist = "weibull"; form = "aft"; (fairly used, **default option**)
- dist = "exponential"; form = "aft";
- dist = "loglogistic"; form = "aft"; (fairly used)
- dist = "lognormal"; form = "aft";
- dist = "loglogistic"; form = "po";

Custom mlr3 parameters

- `discrete` determines the class of the returned survival probability distribution. If `FALSE` (default) continuous probability distributions are returned using distr6::VectorDistribution, otherwise distr6::Matdist (faster to calculate survival measures that require a `distr` prediction type).

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.parametric")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”, “lp”
- Feature Types: “logical”, “integer”, “numeric”, “factor”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **survival**, **pracma**

Parameters

Id	Type	Default	Levels	Range
form	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	-		-
discrete	logical	-	TRUE, FALSE	-

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvParametric`

Methods

Public methods:

- `LearnerSurvParametric$new()`
- `LearnerSurvParametric$clone()`

Method `new()`**:** Creates a new instance of this R6 class. returned `risk` from survivalmodels is hp-style ie higher value => higher risk

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)

bblodfon

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

```
# Define the Learner
learner = mlr3::lrn("surv.parametric")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)
```

```
# Score the predictions
predictions$score()
```

`mlr_learners_surv.pchazard`
Survival PC-Hazard Learner

Description

Logistic-Hazard fits a discrete neural network based on a cross-entropy loss and predictions of a discrete hazard function, also known as Nnet-Survival. 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 lrn():

```
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(32L, 32L)	
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, leakyre
custom_net	untyped	-	
device	untyped	-	
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
sub	integer	10	

Installation

Package 'survivalmodels' is not on CRAN and has to be install from GitHub via `remotes::install_github("RaphaelS1/s`

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

```
lrn("surv.pchazard")
```

`mlr_learners_surv.penalized`
Survival L1 and L2 Penalized Regression Learner

Description

Penalized (L1 and L2) generalized linear models. Calls `penalized::penalized()` from **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 lrn():

```
lrn("surv.penalized")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "numeric", "factor"
- 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

```
# Define the Learner
learner = mlr3::lrn("surv.penalized")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)


# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.priority_lasso`
Survival Priority Lasso Learner

Description

Patient outcome prediction based on multi-omics data taking practitioners' preferences into account. Calls `prioritylasso::prioritylasso()` from **prioritylasso**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.priority_lasso")
```

Meta Information

- Task type: "surv"
- Predict Types: "lp", "response"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: **mlr3**, **mlr3proba**, **prioritylasso**

Parameters

Id	Type	Default	Levels	Range
blocks	untyped	-		-
max.coef	untyped	NULL		-
block1.penalization	logical	TRUE	TRUE, FALSE	-
lambda.type	character	lambda.min	lambda.min, lambda.1se	-
standardize	logical	TRUE	TRUE, FALSE	-
nfolds	integer	5		$[1, \infty)$
foldid	untyped	NULL		-
cvoffset	logical	FALSE	TRUE, FALSE	-
cvoffsetnfolds	integer	10		$[1, \infty)$
return.x	logical	TRUE	TRUE, FALSE	-
handle.missingtestdata	character	-	none, omit.prediction, set.zero, impute.block	-
include.allintercepts	logical	FALSE	TRUE, FALSE	-
use.blocks	untyped	"all"		-
alignment	character	lambda	lambda, fraction	-
alpha	numeric	1		$[0, 1]$
big	numeric	9.9e+35		$(-\infty, \infty)$
devmax	numeric	0.999		$[0, 1]$
dfmax	integer	-		$[0, \infty)$
eps	numeric	1e-06		$[0, 1]$
epsnr	numeric	1e-08		$[0, 1]$
exclude	untyped	-		-
exmx	numeric	250		$(-\infty, \infty)$
fdev	numeric	1e-05		$[0, 1]$
gamma	untyped	-		-
grouped	logical	TRUE	TRUE, FALSE	-
intercept	logical	TRUE	TRUE, FALSE	-
keep	logical	FALSE	TRUE, FALSE	-
lambda	untyped	-		-
lambda.min.ratio	numeric	-		$[0, 1]$
lower.limits	untyped	-Inf		-
maxit	integer	100000		$[1, \infty)$
mnlam	integer	5		$[1, \infty)$
mxit	integer	100		$[1, \infty)$
mxitnr	integer	25		$[1, \infty)$
nlambda	integer	100		$[1, \infty)$
offset	untyped	NULL		-
parallel	logical	FALSE	TRUE, FALSE	-
penalty.factor	untyped	-		-
pmax	integer	-		$[0, \infty)$
pmin	numeric	1e-09		$[0, 1]$
prec	numeric	1e-10		$(-\infty, \infty)$
standardize.response	logical	FALSE	TRUE, FALSE	-
thresh	numeric	1e-07		$[0, \infty)$
trace.it	integer	0		$[0, 1]$
type.gaussian	character	-	covariance, naive	-

type.logistic	character	Newton	Newton, modified.Newton	-
type.multinomial	character	ungrouped	ungrouped, grouped	-
upper.limits	untyped	Inf		-
predict.gamma	numeric	gamma.1se		$(-\infty, \infty)$
relax	logical	FALSE	TRUE, FALSE	-
s	numeric	lambda.1se		$[0, 1]$

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvPriorityLasso`

Methods

Public methods:

- `LearnerSurvPriorityLasso$new()`
- `LearnerSurvPriorityLasso$selected_features()`
- `LearnerSurvPriorityLasso$clone()`

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

Usage:

```
LearnerSurvPriorityLasso$new()
```

Method `selected_features()`**:** Selected features, i.e. those where the coefficient is positive.

Usage:

```
LearnerSurvPriorityLasso$selected_features()
```

Returns: `character()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

HarutyunyanLiana

References

Simon K, Vindi J, Roman H, Tobias H, Anne-Laure B (2018). "Priority-Lasso: a simple hierarchical approach to the prediction of clinical outcome using multi-omics data." *BMC Bioinformatics*, **19**. doi:10.1186/s1285901823446.

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

```
lrn("surv.priority_lasso")
```

---

mlr_learners_surv.ranger

Ranger Survival Learner

Description

Random survival forest. Calls `ranger::ranger()` from package **ranger**.

Custom mlr3 parameters

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

Initial parameter values

- `num.threads` is initialized to 1 to avoid conflicts with parallelization via **future**.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.ranger")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "distr"
- Feature Types: "logical", "integer", "numeric", "character", "factor", "ordered"
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **ranger**

Parameters

Id	Type	Default	Levels	Range
alpha	numeric	0.5		$(-\infty, \infty)$
always.split.variables	untyped	-		-
holdout	logical	FALSE	TRUE, FALSE	-
importance	character	-	none, impurity, impurity_corrected, permutation	-
keep.inbag	logical	FALSE	TRUE, FALSE	-
max.depth	integer	NULL		$[0, \infty)$
min.node.size	integer	5		$[1, \infty)$
minprop	numeric	0.1		$(-\infty, \infty)$
mtry	integer	-		$[1, \infty)$
mtry.ratio	numeric	-		$[0, 1]$
num.random.splits	integer	1		$[1, \infty)$
num.threads	integer	1		$[1, \infty)$
num.trees	integer	500		$[1, \infty)$
oob.error	logical	TRUE	TRUE, FALSE	-
regularization.factor	untyped	1		-
regularization.usedepth	logical	FALSE	TRUE, FALSE	-
replace	logical	TRUE	TRUE, FALSE	-
respect.unordered.factors	character	ignore	ignore, order, partition	-
sample.fraction	numeric	-		$[0, 1]$
save.memory	logical	FALSE	TRUE, FALSE	-
scale.permutation.importance	logical	FALSE	TRUE, FALSE	-
seed	integer	NULL		$(-\infty, \infty)$
split.select.weights	numeric	-		$[0, 1]$
splitrule	character	logrank	logrank, extratrees, C, maxstat	-
verbose	logical	TRUE	TRUE, FALSE	-
write.forest	logical	TRUE	TRUE, FALSE	-
min.bucket	integer	3		$(-\infty, \infty)$
time.interest	integer	NULL		$[1, \infty)$
node.stats	logical	FALSE	TRUE, FALSE	-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvRanger`

Methods

Public methods:

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

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

Usage:

```
LearnerSurvRanger$new()
```

Method `importance()`**:** The importance scores are extracted from the model slot `variable.importance`.

Usage:

```
LearnerSurvRanger$importance()
```

Returns: Named `numeric()`.

Method `oob_error()`**:** The out-of-bag error is extracted from the model slot `prediction.error`.

Usage:

```
LearnerSurvRanger$oob_error()
```

Returns: `numeric(1)`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Author(s)

be-marc

References

Wright, N. M, Ziegler, Andreas (2017). "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software*, **77**(1), 1–17. doi:10.18637/jss.v077.i01.

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

```
# Define the Learner
learner = mlr3::lrn("surv.ranger")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.rfsrc`

Survival Random Forest SRC Learner

Description

Random survival forest. Calls `randomForestSRC::rfsrc()` from **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 lrn():

```
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		$[1, \infty)$
save.memory	logical	FALSE	TRUE, FALSE	-
perf.type	character	-	none	-
case.depth	logical	FALSE	TRUE, FALSE	-

Custom mlr3 parameters

- `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.
- `cores`: This value is set as the option `rf.cores` during training and is set to 1 by default.

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

```
# Define the Learner
learner = mlr3::lrn("surv.rfsrc")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)
```

```
print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.svm` *Survival Support Vector Machine Learner*

Description

Survival support vector machine. Calls `survivalsvm::survivalsvm()` from **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 lrn():

```
lrn("surv.svm")
```

Meta Information

- Task type: "surv"
- Predict Types: "crank", "response"
- Feature Types: "logical", "integer", "numeric", "character", "factor"
- 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

```
set.seed(123)
# Define the Learner and set parameter values
learner = lrn("surv.svm", gamma.mu = 0.1)
print(learner)

# Define a Task
task = mlr3::tsk("rats")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

# print the model
print(learner$model)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

---

mlr_learners_surv.xgboost.aft

Extreme Gradient Boosting AFT Survival Learner

Description

eXtreme Gradient Boosting regression using an **Accelerated Failure Time** objective. Calls `xgboost::xgb.train()` from package **xgboost** with `objective` set to `survival:aft` and `eval_metric` to `aft-nloglik`.

Details

This learner returns three prediction types:

1. `response`: the estimated survival time T for each test observation.
2. `lp`: a vector of linear predictors (relative risk scores), one per observation, estimated as $-log(T)$. Higher survival time denotes lower risk.
3. `crank`: same as `lp`.

Initial parameter values

- `nrounds` is initialized to 1000.
- `nthread` is initialized to 1 to avoid conflicts with parallelization via **future**.
- `verbose` is initialized to 0.

Early stopping

Early stopping can be used to find the optimal number of boosting rounds. The `early_stopping_set` parameter controls which set is used to monitor the performance. By default, `early_stopping_set = "none"` which disables early stopping. Set `early_stopping_set = "test"` to monitor the performance of the model on the test set while training. The test set for early stopping can be set with the `"test"` row role in the mlr3::Task. Additionally, the range must be set in which the performance must increase with `early_stopping_rounds` and the maximum number of boosting rounds with `nrounds`. While resampling, the test set is automatically applied from the mlr3::Resampling. Not that using the test set for early stopping can potentially bias the performance scores.

Early stopping can be used to find the optimal number of boosting rounds. The `early_stopping_set` parameter controls which set is used to monitor the performance. By default, `early_stopping_set = "none"` which disables early stopping. Set `early_stopping_set = "test"` to monitor the performance of the model on the test set while training. The test set for early stopping can be set with the `"test"` row role in the mlr3::Task. Additionally, the range must be set in which the performance must increase with `early_stopping_rounds` and the maximum number of boosting rounds with `nrounds`. While resampling, the test set is automatically applied from the mlr3::Resampling. Not that using the test set for early stopping can potentially bias the performance scores.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.xgboost.aft")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “lp”, “response”
- Feature Types: “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **xgboost**

Parameters

Id	Type	Default	Levels	Range
aft_loss_distribution	character	normal	normal, logistic, extreme	-
aft_loss_distribution_scale	numeric	-		$(-\infty, \infty)$
alpha	numeric	0		$[0, \infty)$
base_score	numeric	0.5		$(-\infty, \infty)$
booster	character	gbtree	gbtree, gblinear, dart	-
callbacks	untyped	list()		-
colsample_bylevel	numeric	1		$[0, 1]$
colsample_bynode	numeric	1		$[0, 1]$
colsample_bytree	numeric	1		$[0, 1]$
disable_default_eval_metric	logical	FALSE	TRUE, FALSE	-
early_stopping_rounds	integer	NULL		$[1, \infty)$
eta	numeric	0.3		$[0, 1]$
feature_selector	character	cyclic	cyclic, shuffle, random, greedy, thrifty	-
feval	untyped	NULL		-
gamma	numeric	0		$[0, \infty)$
grow_policy	character	depthwise	depthwise, lossguide	-
interaction_constraints	untyped	-		-
iterationrange	untyped	-		-
lambda	numeric	1		$[0, \infty)$
lambda_bias	numeric	0		$[0, \infty)$
max_bin	integer	256		$[2, \infty)$
max_delta_step	numeric	0		$[0, \infty)$
max_depth	integer	6		$[0, \infty)$
max_leaves	integer	0		$[0, \infty)$
maximize	logical	NULL	TRUE, FALSE	-
min_child_weight	numeric	1		$[0, \infty)$
missing	numeric	NA		$(-\infty, \infty)$
monotone_constraints	integer	0		$[-1, 1]$
normalize_type	character	tree	tree, forest	-
nrounds	integer	-		$[1, \infty)$
nthread	integer	1		$[1, \infty)$
ntreelimit	integer	-		$[1, \infty)$
num_parallel_tree	integer	1		$[1, \infty)$
one_drop	logical	FALSE	TRUE, FALSE	-
print_every_n	integer	1		$[1, \infty)$
process_type	character	default	default, update	-
rate_drop	numeric	0		$[0, 1]$
refresh_leaf	logical	TRUE	TRUE, FALSE	-
sampling_method	character	uniform	uniform, gradient_based	-
sample_type	character	uniform	uniform, weighted	-
save_name	untyped	-		-
save_period	integer	-		$[0, \infty)$
scale_pos_weight	numeric	1		$(-\infty, \infty)$
seed_per_iteration	logical	FALSE	TRUE, FALSE	-
skip_drop	numeric	0		$[0, 1]$

strict_shape	logical	FALSE	TRUE, FALSE	-
subsample	numeric	1		$[0, 1]$
top_k	integer	0		$[0, \infty)$
tree_method	character	auto	auto, exact, approx, hist, gpu_hist	-
tweedie_variance_power	numeric	1.5		$[1, 2]$
updater	untyped	-		-
verbose	integer	1		$[0, 2]$
watchlist	untyped	NULL		-
xgb_model	untyped	-		-
device	untyped	-		-

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvXgboostAFT`

Active bindings

`internal_valid_scores` The last observation of the validation scores for all metrics. Extracted from `model$evaluation_log`

`internal_tuned_values` Returns the early stopped iterations if `early_stopping_rounds` was set during training.

`validate` How to construct the internal validation data. This parameter can be either `NULL`, a ratio, `"test"`, or `"predefined"`.

Methods

Public methods:

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

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

Usage:

```
LearnerSurvXgboostAFT$new()
```

Method `importance()`**:** The importance scores are calculated with `xgboost::xgb.importance()`.

Usage:

```
LearnerSurvXgboostAFT$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

To compute on GPUs, you first need to compile **xgboost** yourself and link against CUDA. See https://xgboost.readthedocs.io/en/stable/build.html#building-with-gpu-support.

Author(s)

bblodfon

References

Chen, Tianqi, Guestrin, Carlos (2016). "Xgboost: A scalable tree boosting system." In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 785–794. ACM. doi:10.1145/2939672.2939785.

Avinash B, Hyunsu C, Toby H (2022). "Survival Regression with Accelerated Failure Time Model in XGBoost." *Journal of Computational and Graphical Statistics*. ISSN 15372715, doi:10.1080/10618600.2022.2067548.

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

```
# Define the Learner
learner = mlr3::lrn("surv.xgboost.aft")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
```

```
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

`mlr_learners_surv.xgboost.cox`

Extreme Gradient Boosting Cox Survival Learner

Description

eXtreme Gradient Boosting regression using a **Cox Proportional Hazards** objective. Calls `xgboost::xgb.train()` from package **xgboost** with `objective` set to `survival:cox` and `eval_metric` to `cox-nloglik`.

Details

Three types of prediction are returned for this learner:

1. `lp`: a vector of linear predictors (relative risk scores), one per observation.
2. `crank`: same as `lp`.
3. `distr`: a survival matrix in two dimensions, where observations are represented in rows and time points in columns. By default, the Breslow estimator is used via `mlr3proba::breslow()`.

Initial parameter values

- `nrounds` is initialized to 1000.
- `nthread` is initialized to 1 to avoid conflicts with parallelization via **future**.
- `verbose` is initialized to 0.

Dictionary

This Learner can be instantiated via lrn():

```
lrn("surv.xgboost.cox")
```

Meta Information

- Task type: “surv”
- Predict Types: “crank”, “distr”, “lp”
- Feature Types: “integer”, “numeric”
- Required Packages: **mlr3**, **mlr3proba**, **mlr3extralearners**, **xgboost**

Parameters

Id	Type	Default	Levels	Range
alpha	numeric	0		$[0, \infty)$
base_score	numeric	0.5		$(-\infty, \infty)$
booster	character	gbtree	gbtree, gblinear, dart	-
callbacks	untyped	list()		-
colsample_bylevel	numeric	1		$[0, 1]$
colsample_bynode	numeric	1		$[0, 1]$
colsample_bytree	numeric	1		$[0, 1]$
disable_default_eval_metric	logical	FALSE	TRUE, FALSE	-
early_stopping_rounds	integer	NULL		$[1, \infty)$
eta	numeric	0.3		$[0, 1]$
feature_selector	character	cyclic	cyclic, shuffle, random, greedy, thrifty	-
feval	untyped	NULL		-
gamma	numeric	0		$[0, \infty)$
grow_policy	character	depthwise	depthwise, lossguide	-
interaction_constraints	untyped	-		-
iterationrange	untyped	-		-
lambda	numeric	1		$[0, \infty)$
lambda_bias	numeric	0		$[0, \infty)$
max_bin	integer	256		$[2, \infty)$
max_delta_step	numeric	0		$[0, \infty)$
max_depth	integer	6		$[0, \infty)$
max_leaves	integer	0		$[0, \infty)$
maximize	logical	NULL	TRUE, FALSE	-
min_child_weight	numeric	1		$[0, \infty)$
missing	numeric	NA		$(-\infty, \infty)$
monotone_constraints	integer	0		$[-1, 1]$
normalize_type	character	tree	tree, forest	-
nrounds	integer	-		$[1, \infty)$
nthread	integer	1		$[1, \infty)$
num_parallel_tree	integer	1		$[1, \infty)$
one_drop	logical	FALSE	TRUE, FALSE	-
print_every_n	integer	1		$[1, \infty)$
process_type	character	default	default, update	-
rate_drop	numeric	0		$[0, 1]$
refresh_leaf	logical	TRUE	TRUE, FALSE	-
sampling_method	character	uniform	uniform, gradient_based	-
sample_type	character	uniform	uniform, weighted	-
save_name	untyped	-		-
save_period	integer	-		$[0, \infty)$
scale_pos_weight	numeric	1		$(-\infty, \infty)$
seed_per_iteration	logical	FALSE	TRUE, FALSE	-
skip_drop	numeric	0		$[0, 1]$
strict_shape	logical	FALSE	TRUE, FALSE	-
subsample	numeric	1		$[0, 1]$
top_k	integer	0		$[0, \infty)$
tree_method	character	auto	auto, exact, approx, hist, gpu_hist	-
tweedie_variance_power	numeric	1.5		$[1, 2]$

updater	untyped	-	-
verbose	integer	1	$[0, 2]$
watchlist	untyped	NULL	-
xgb_model	untyped	-	-
device	untyped	-	-

Early stopping

Early stopping can be used to find the optimal number of boosting rounds. The `early_stopping_set` parameter controls which set is used to monitor the performance. By default, `early_stopping_set = "none"` which disables early stopping. Set `early_stopping_set = "test"` to monitor the performance of the model on the test set while training. The test set for early stopping can be set with the `"test"` row role in the mlr3::Task. Additionally, the range must be set in which the performance must increase with `early_stopping_rounds` and the maximum number of boosting rounds with `nrounds`. While resampling, the test set is automatically applied from the mlr3::Resampling. Not that using the test set for early stopping can potentially bias the performance scores.

Super classes

`mlr3::Learner` -> `mlr3proba::LearnerSurv` -> `LearnerSurvXgboostCox`

Active bindings

`internal_valid_scores` The last observation of the validation scores for all metrics. Extracted from `model$evaluation_log`

`internal_tuned_values` Returns the early stopped iterations if `early_stopping_rounds` was set during training.

`validate` How to construct the internal validation data. This parameter can be either `NULL`, a ratio, `"test"`, or `"predefined"`.

Methods

Public methods:

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

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

Usage:

```
LearnerSurvXgboostCox$new()
```

Method `importance()`**:** The importance scores are calculated with `xgboost::xgb.importance()`.

Usage:

```
LearnerSurvXgboostCox$importance()
```

Returns: Named `numeric()`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

To compute on GPUs, you first need to compile **xgboost** yourself and link against CUDA. See https://xgboost.readthedocs.io/en/stable/build.html#building-with-gpu-support.

Author(s)

bblodfon

References

Chen, Tianqi, Guestrin, Carlos (2016). "Xgboost: A scalable tree boosting system." In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 785–794. ACM. doi:10.1145/2939672.2939785.

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

```
# Define the Learner
learner = mlr3::lrn("surv.xgboost.cox")
print(learner)

# Define a Task
task = mlr3::tsk("grace")

# Create train and test set
ids = mlr3::partition(task)

# Train the learner on the training ids
```

```
learner$train(task, row_ids = ids$train)

print(learner$model)
print(learner$importance)

# Make predictions for the test rows
predictions = learner$predict(task, row_ids = ids$test)

# Score the predictions
predictions$score()
```

Index