Package 'mlr3resampling'

April 16, 2024

Type Package

Title Resampling Algorithms for 'mlr3' Framework

Version 2024.4.14

Description A supervised learning algorithm inputs a train set,
and outputs a prediction function, which can be used on a test set.
If each data point belongs to a group
(such as geographic region, year, etc), then
how do we know if it is possible to train on one group, and predict
accurately on another group? Cross-validation can be used to determine
the extent to which this is possible, by first assigning fold IDs from
1 to K to all data (possibly using stratification, usually by group
and label). Then we loop over test sets (group/fold combinations),
train sets (same group, other groups, all groups), and compute
test/prediction accuracy for each combination. Comparing
test/prediction accuracy between same and other, we can determine the
extent to which it is possible (perfect if same/other have similar
test accuracy for each group; other is usually somewhat less accurate
than same; other can be just as bad as featureless baseline when the
groups have different patterns).
For more information,
<https://tdhock.github.io/blog/2023/R-gen-new-subsets/>
describes the method in depth.
How many train samples are required to get accurate predictions on a
test set? Cross-validation can be used to answer this question, with
variable size train sets.

License GPL-3

URL https://github.com/tdhock/mlr3resampling

BugReports https://github.com/tdhock/mlr3resampling/issues

Imports data.table, R6, checkmate, paradox, mlr3, mlr3misc

Suggests animint2, mlr3tuning, lgr, future, testthat, knitr,
rmarkdown, nc, rpart

VignetteBuilder knitr

NeedsCompilation no

Author Toby Hocking [aut, cre] (<https://orcid.org/0000-0002-3146-0865>),
Michel Lang [ctb] (<https://orcid.org/0000-0001-9754-0393>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Bernd Bischl [ctb] (<https://orcid.org/0000-0001-6002-6980>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Jakob Richter [ctb] (<https://orcid.org/0000-0003-4481-5554>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Patrick Schratz [ctb] (<https://orcid.org/0000-0003-0748-6624>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Giuseppe Casalicchio [ctb] (<https://orcid.org/0000-0001-5324-5966>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Stefan Coors [ctb] (<https://orcid.org/0000-0002-7465-2146>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Quay Au [ctb] (<https://orcid.org/0000-0002-5252-8902>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Martin Binder [ctb],
Florian Pfisterer [ctb] (<https://orcid.org/0000-0001-8867-762X>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Raphael Sonabend [ctb] (<https://orcid.org/0000-0001-9225-4654>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Lennart Schneider [ctb] (<https://orcid.org/0000-0003-4152-5308>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Marc Becker [ctb] (<https://orcid.org/0000-0002-8115-0400>, Author of mlr3 when Resampling/ResamplingCV was copied/modified),
Sebastian Fischer [ctb] (<https://orcid.org/0000-0002-9609-3197>, Author of mlr3 when Resampling/ResamplingCV was copied/modified)

Maintainer Toby Hocking <toby.hocking@r-project.org>

Repository CRAN

Date/Publication 2024-04-16 15:50:02 UTC

R topics documented:

`AZtrees` *Arizona Trees*

Description

Classification data set with polygons (groups which should not be split in CV) and subsets (region3 or region4).

Usage

```
data("AZtrees")
```

Format

A data frame with 5956 observations on the following 25 variables.

`region3` a character vector

`region4` a character vector

`polygon` a numeric vector

`y` a character vector

`SAMPLE_1` a numeric vector

`SAMPLE_2` a numeric vector

`SAMPLE_3` a numeric vector

`SAMPLE_4` a numeric vector

`SAMPLE_5` a numeric vector

`SAMPLE_6` a numeric vector

`SAMPLE_7` a numeric vector

`SAMPLE_8` a numeric vector

`SAMPLE_9` a numeric vector

`SAMPLE_10` a numeric vector

`SAMPLE_11` a numeric vector

`SAMPLE_12` a numeric vector

`SAMPLE_13` a numeric vector

`SAMPLE_14` a numeric vector

`SAMPLE_15` a numeric vector

`SAMPLE_16` a numeric vector

`SAMPLE_17` a numeric vector

`SAMPLE_18` a numeric vector

`SAMPLE_19` a numeric vector

`SAMPLE_20` a numeric vector

`SAMPLE_21` a numeric vector

Source

Paul Nelson Arellano, paul.arellano@nau.edu

Examples

```
data(AZtrees)
task.obj <- mlr3::TaskClassif$new("AZtrees3", AZtrees, target="y")
task.obj$col_roles$feature <- grep("SAMPLE", names(AZtrees), value=TRUE)
task.obj$col_roles$group <- "polygon"
task.obj$col_roles$subset <- "region3"
str(task.obj)
same_other_sizes_cv <- mlr3resampling::ResamplingSameOtherSizesCV$new()
same_other_sizes_cv$instantiate(task.obj)
same_other_sizes_cv$instance$iteration.dt
```

ResamplingSameOtherCV *Resampling for comparing training on same or other groups*

Description

ResamplingSameOtherCV defines how a task is partitioned for resampling, for example in resample() or benchmark().

Resampling objects can be instantiated on a Task, which should define at least one group variable.

After instantiation, sets can be accessed via $train_set(i) and $test_set(i), respectively.

Details

A supervised learning algorithm inputs a train set, and outputs a prediction function, which can be used on a test set. If each data point belongs to a group (such as geographic region, year, etc), then how do we know if it is possible to train on one group, and predict accurately on another group? Cross-validation can be used to determine the extent to which this is possible, by first assigning fold IDs from 1 to K to all data (possibly using stratification, usually by group and label). Then we loop over test sets (group/fold combinations), train sets (same group, other groups, all groups), and compute test/prediction accuracy for each combination. Comparing test/prediction accuracy between same and other, we can determine the extent to which it is possible (perfect if same/other have similar test accuracy for each group; other is usually somewhat less accurate than same; other can be just as bad as featureless baseline when the groups have different patterns).

Stratification

ResamplingSameOtherCV supports stratified sampling. The stratification variables are assumed to be discrete, and must be stored in the Task with column role "stratum". In case of multiple stratification variables, each combination of the values of the stratification variables forms a stratum.

Grouping

ResamplingSameOtherCV supports grouping of observations. The grouping variable is assumed to be discrete, and must be stored in the Task with column role "group".

The number of cross-validation folds K should be defined as the fold parameter.

In each group, there will be about an equal number of observations assigned to each of the K folds. The assignments are stored in `$instance$id.dt`. The train/test splits are defined by all possible combinations of test group, test fold, and train groups (same/other/all). The splits are stored in `$instance$iteration.dt`.

Methods

Public methods:

- `Resampling$new()`
- `Resampling$train_set()`
- `Resampling$test_set()`

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

Usage:

```
Resampling$new(
  id,
  param_set = ps(),
  duplicated_ids = FALSE,
  label = NA_character_,
  man = NA_character_
)
```

Arguments:

`id` (`character(1)`)
: Identifier for the new instance.

`param_set` (paradox::ParamSet)
: Set of hyperparameters.

`duplicated_ids` (`logical(1)`)
: Set to `TRUE` if this resampling strategy may have duplicated row ids in a single training set or test set.
Note that this object is typically constructed via a derived classes, e.g. ResamplingCV or ResamplingHoldout.

`label` (`character(1)`)
: Label for the new instance.

`man` (`character(1)`)
: String in the format `[pkg]::[topic]` pointing to a manual page for this object. The referenced help package can be opened via method `$help()`.

Method `train_set()`**:** Returns the row ids of the i-th training set.

Usage:

```
Resampling$train_set(i)
```

Arguments:

`i` (`integer(1)`)
: Iteration.

Returns: (`integer()`) of row ids.

Method `test_set()`**:** Returns the row ids of the i-th test set.

Usage:

```
Resampling$test_set(i)
```

Arguments:

`i` (`integer(1)`)
Iteration.

Returns: (`integer()`) of row ids.

See Also

- Blog post https://tdhock.github.io/blog/2023/R-gen-new-subsets/
- Package **mlr3** for standard `Resampling`, which does not support comparing train on same or other groups.
- `score` and Simulations vignette for more detailed examples.

Examples

```
same_other <- mlr3resampling::ResamplingSameOtherCV$new()
same_other$param_set$values$folds <- 5
```

`ResamplingSameOtherSizesCV`
Resampling for comparing train subsets and sizes

Description

`ResamplingSameOtherSizesCV` defines how a task is partitioned for resampling, for example in `resample()` or `benchmark()`.

Resampling objects can be instantiated on a `Task`, which should define at least one group variable.

After instantiation, sets can be accessed via `$train_set(i)` and `$test_set(i)`, respectively.

Details

A supervised learning algorithm inputs a train set, and outputs a prediction function, which can be used on a test set. If each data point belongs to a group (such as geographic region, year, etc), then how do we know if it is possible to train on one group, and predict accurately on another group? Cross-validation can be used to determine the extent to which this is possible, by first assigning fold IDs from 1 to K to all data (possibly using stratification, usually by group and label). Then we loop over test sets (group/fold combinations), train sets (same group, other groups, all groups), and compute test/prediction accuracy for each combination. Comparing test/prediction accuracy between same and other, we can determine the extent to which it is possible (perfect if same/other have similar test accuracy for each group; other is usually somewhat less accurate than same; other can be just as bad as featureless baseline when the groups have different patterns).

This class has more parameters/potential applications than `ResamplingSameOtherCV` and `ResamplingVariableSizeTrainC`
which are older and should only be preferred for visualization purposes.

Stratification

ResamplingSameOtherSizesCV supports stratified sampling. The stratification variables are assumed to be discrete, and must be stored in the Task with column role "stratum". In case of multiple stratification variables, each combination of the values of the stratification variables forms a stratum.

Grouping

ResamplingSameOtherSizesCV supports grouping of observations. The grouping variable is assumed to be discrete, and must be stored in the Task with column role "group".

Subsets

ResamplingSameOtherSizesCV supports training on different subsets of observations. The subset variable is assumed to be discrete, and must be stored in the Task with column role "subset".

Parameters

The number of cross-validation folds K should be defined as the fold parameter, default 3.

The number of random seeds for down-sampling should be defined as the seeds parameter, default 1.

The ratio for down-sampling should be defined as the ratio parameter, default 0.5. The min size of same and other sets is repeatedly multiplied by this ratio, to obtain smaller sample sizes.

The number of down-sampling sizes/multiplications should be defined as the sizes parameter, which can also take two special values: default -1 means no down-sampling at all, and 0 means only down-sampling to the sizes of the same/other sets.

The ignore_subset parameter should be either TRUE or FALSE (default), whether to ignore the subset role. TRUE only creates splits for same subset (even if task defines subset role), and is useful for subtrain/validation splits (hyper-parameter learning). Note that this feature will work on a task with stratum and group roles (unlike ResamplingCV).

In each subset, there will be about an equal number of observations assigned to each of the K folds. The train/test splits are defined by all possible combinations of test subset, test fold, train subsets (same/other/all), down-sampling sizes, and random seeds. The splits are stored in $instance$iteration.dt.

Methods

Public methods:

- Resampling$new()
- Resampling$train_set()
- Resampling$test_set()

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

Usage:

```
Resampling$new(
  id,
  param_set = ps(),
```

```
  duplicated_ids = FALSE,
  label = NA_character_,
  man = NA_character_
)
```

Arguments:

`id (character(1))`
: Identifier for the new instance.

`param_set (paradox::ParamSet)`
: Set of hyperparameters.

`duplicated_ids (logical(1))`
: Set to `TRUE` if this resampling strategy may have duplicated row ids in a single training set or test set.
Note that this object is typically constructed via a derived classes, e.g. ResamplingCV or ResamplingHoldout.

`label (character(1))`
: Label for the new instance.

`man (character(1))`
: String in the format `[pkg]::[topic]` pointing to a manual page for this object. The referenced help package can be opened via method `$help()`.

Method `train_set()`**:** Returns the row ids of the i-th training set.

Usage:

```
Resampling$train_set(i)
```

Arguments:

`i (integer(1))`
: Iteration.

Returns: (`integer()`) of row ids.

Method `test_set()`**:** Returns the row ids of the i-th test set.

Usage:

```
Resampling$test_set(i)
```

Arguments:

`i (integer(1))`
: Iteration.

Returns: (`integer()`) of row ids.

See Also

- Blog post https://tdhock.github.io/blog/2023/R-gen-new-subsets/
- Package **mlr3** for standard `Resampling`, which does not support comparing train on same or other groups.
- `score` and Simulations vignette for more detailed examples.

Examples

```
same_other_sizes <- mlr3resampling::ResamplingSameOtherSizesCV$new()
same_other_sizes$param_set$values$folds <- 5
```

ResamplingVariableSizeTrainCV *Resampling for comparing training on same or other groups*

Description

ResamplingVariableSizeTrainCV defines how a task is partitioned for resampling, for example in resample() or benchmark().

Resampling objects can be instantiated on a Task.

After instantiation, sets can be accessed via $train_set(i) and $test_set(i), respectively.

Details

A supervised learning algorithm inputs a train set, and outputs a prediction function, which can be used on a test set. How many train samples are required to get accurate predictions on a test set? Cross-validation can be used to answer this question, with variable size train sets.

Stratification

ResamplingVariableSizeTrainCV supports stratified sampling. The stratification variables are assumed to be discrete, and must be stored in the Task with column role "stratum". In case of multiple stratification variables, each combination of the values of the stratification variables forms a stratum.

Grouping

ResamplingVariableSizeTrainCV does not support grouping of observations.

Hyper-parameters

The number of cross-validation folds should be defined as the fold parameter.

For each fold ID, the corresponding observations are considered the test set, and a variable number of other observations are considered the train set.

The random_seeds parameter controls the number of random orderings of the train set that are considered.

For each random order of the train set, the min_train_data parameter controls the size of the smallest stratum in the smallest train set considered.

To determine the other train set sizes, we use an equally spaced grid on the log scale, from min_train_data to the largest train set size (all data not in test set). The number of train set sizes in this grid is determined by the train_sizes parameter.

Methods

Public methods:

- `Resampling$new()`
- `Resampling$train_set()`
- `Resampling$test_set()`

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

Usage:

```
Resampling$new(
  id,
  param_set = ps(),
  duplicated_ids = FALSE,
  label = NA_character_,
  man = NA_character_
)
```

Arguments:

`id` (`character(1)`)
: Identifier for the new instance.

`param_set` (paradox::ParamSet)
: Set of hyperparameters.

`duplicated_ids` (`logical(1)`)
: Set to `TRUE` if this resampling strategy may have duplicated row ids in a single training set or test set.
Note that this object is typically constructed via a derived classes, e.g. ResamplingCV or ResamplingHoldout.

`label` (`character(1)`)
: Label for the new instance.

`man` (`character(1)`)
: String in the format `[pkg]::[topic]` pointing to a manual page for this object. The referenced help package can be opened via method `$help()`.

Method `train_set()`**:** Returns the row ids of the i-th training set.

Usage:

```
Resampling$train_set(i)
```

Arguments:

`i` (`integer(1)`)
: Iteration.

Returns: (`integer()`) of row ids.

Method `test_set()`**:** Returns the row ids of the i-th test set.

Usage:

```
Resampling$test_set(i)
```

Arguments:

`i` (`integer(1)`)
: Iteration.

Returns: (`integer()`) of row ids.

Examples

```
(var_sizes <- mlr3resampling::ResamplingVariableSizeTrainCV$new())
```

---

score *Score benchmark results*

Description

Computes a data table of scores.

Usage

```
score(bench.result, ...)
```

Arguments

bench.result	Output of benchmark().
...	Additional arguments to pass to bench.result$score, for example measures.

Value

data table with scores.

Author(s)

Toby Dylan Hocking

Examples

```
N <- 100
library(data.table)
set.seed(1)
reg.dt <- data.table(
  x=runif(N, -2, 2),
  person=rep(1:2, each=0.5*N))
reg.pattern.list <- list(
  easy=function(x, person)x^2,
  impossible=function(x, person)(x^2+person*3)*(-1)^person)
reg.task.list <- list()
for(pattern in names(reg.pattern.list)){
  f <- reg.pattern.list[[pattern]]
  yname <- paste0("y_",pattern)
  reg.dt[, (yname) := f(x,person)+rnorm(N, sd=0.5)][]
  task.dt <- reg.dt[, c("x","person",yname), with=FALSE]
  task.obj <- mlr3::TaskRegr$new(
    pattern, task.dt, target=yname)
  task.obj$col_roles$stratum <- "person"
  task.obj$col_roles$subset <- "person"
```

```
  reg.task.list[[pattern]] <- task.obj
}
same_other <- mlr3resampling::ResamplingSameOtherSizesCV$new()
reg.learner.list <- list(
  mlr3::LearnerRegrFeatureless$new())
if(requireNamespace("rpart")){
  reg.learner.list$rpart <- mlr3::LearnerRegrRpart$new()
}
(bench.grid <- mlr3::benchmark_grid(
  reg.task.list,
  reg.learner.list,
  same_other))
bench.result <- mlr3::benchmark(bench.grid)
bench.score <- mlr3resampling::score(bench.result)
if(require(animint2)){
  ggplot()+
    geom_point(aes(
      regr.mse, train.subsets, color=algorithm),
      shape=1,
      data=bench.score)+
    facet_grid(
      test.subset ~ task_id,
      labeller=label_both,
      scales="free")+
    scale_x_log10()
}
```

Index