Package 'mlr3filters'

December 8, 2019

Title Filter Based Feature Selection for 'mlr3'

Version 0.1.1

Description Extends 'mlr3' with filter methods for feature selection. Besides standalone filter methods built-in methods of any machine-learning algorithm are supported. Partial scoring of multivariate filter methods is supported.

License LGPL-3

URL https://mlr3filters.mlr-org.com, https://github.com/mlr-org/mlr3filters

BugReports https://github.com/mlr-org/mlr3filters/issues

Depends R (>= 3.1.0)

Imports backports, checkmate, data.table, mlr3 (>= 0.1.2), mlr3misc, paradox, R6

Suggests care, FSelectorRcpp, lgr, mlr3learners, mlr3measures, praznik, rpart, testthat

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.0.2

Collate 'Filter.R' 'mlr_filters.R' 'FilterAUC.R' 'FilterAnova.R' 'FilterCMIM.R' 'FilterCarScore.R' 'FilterCorrelation.R' 'FilterDISR.R' 'FilterImportance.R' 'FilterInformationGain.R' 'FilterJMI.R' 'FilterJMIM.R' 'FilterKruskalTest.R' 'FilterMIM.R' 'FilterMRMR.R' 'FilterNJMIM.R' 'FilterPerformance.R' 'FilterVariance.R' 'flt.R' 'reexports.R' 'zzz.R'

Author Patrick Schratz [aut, cre] (<https://orcid.org/0000-0003-0748-6624>), Michel Lang [aut] (<https://orcid.org/0000-0001-9754-0393>), Bernd Bischl [aut] (<https://orcid.org/0000-0001-6002-6980>)

Maintainer Patrick Schratz <patrick.schratz@gmail.com>

Repository CRAN

Date/Publication 2019-12-08 13:40:02 UTC

R topics documented:

mlr3filters-package . . . 2
Filter . . . 3
FilterAnova . . . 4
FilterAUC . . . 5
FilterCarScore . . . 6
FilterCMIM . . . 7
FilterCorrelation . . . 7
FilterDISR . . . 8
FilterImportance . . . 9
FilterInformationGain . . . 10
FilterJMI . . . 11
FilterJMIM . . . 11
FilterKruskalTest . . . 12
FilterMIM . . . 13
FilterMRMR . . . 14
FilterNJMIM . . . 14
FilterPerformance . . . 15
FilterVariance . . . 16
flt . . . 17
mlr_filters . . . 17

Index **19**

`mlr3filters-package` *mlr3filters: Filter Based Feature Selection for 'mlr3'*

Description

Extends 'mlr3' with filter methods for feature selection. Besides standalone filter methods built-in methods of any machine-learning algorithm are supported. Partial scoring of multivariate filter methods is supported.

Author(s)

Maintainer: Patrick Schratz <patrick.schratz@gmail.com> (ORCID)

Authors:

- Michel Lang <michellang@gmail.com> (ORCID)
- Bernd Bischl <bernd_bischl@gmx.net> (ORCID)

See Also

Useful links:

- https://mlr3filters.mlr-org.com
- https://github.com/mlr-org/mlr3filters
- Report bugs at https://github.com/mlr-org/mlr3filters/issues

`Filter` *Filter Base Class*

Description

Base class for filters. Predefined filters are stored in the dictionary mlr_filters. A Filter calculates a score for each feature of a task. Important features get a large value and unimportant features get a small value. Note that filter scores may also be negative.

Format

R6::R6Class object.

Construction

```
f = Filter$new(id, task_type, param_set, feature_types, packages)
```

- `id` :: `character(1)`
  Identifier for the filter.
- `task_type` :: `character()`
  Types of the task the filter can operator on. E.g., `"classif"` or `"regr"`.
- `param_set` :: paradox::ParamSet
  Set of hyperparameters.
- `feature_types` :: `character()`
  Feature types the filter operates on. Must be a subset of `mlr_reflections$task_feature_types`.
- `task_properties` :: `character()`
  Required task properties, see mlr3::Task. Must be a subset of `mlr_reflections$task_properties`.
- `packages` :: `character()`
  Set of required packages. Note that these packages will be loaded via `requireNamespace()`, and are not attached.

Fields

All arguments passed to the constructor are available as fields, and additionally:

- `scores` :: named `numeric()`
  Stores the calculated filter score values as named numeric vector. The vector is sorted in decreasing order with possible `NA` values last. Tied values (this includes `NA` values) appear in a random, non-deterministic order.

Methods

- `calculate(task,nfeat = NULL)`
  (mlr3::Task, `integer(1)`) -> `self`
  Calculates the filter score values for the provided mlr3::Task and stores them in field `scores`. `nfeat` determines the minimum number of features to score (see "Partial Scoring"), and defaults to the number of features in `task`. Loads required packages and then calls $calculate_internal(). If the task has no rows, each feature gets the score `NA`.

- `calculate_internal(task,nfeat)`
  (mlr3::Task, `integer(1)`) -> named `numeric()`
  Internal worker function. Each child class muss implement this method. Takes a task and the minimum number of features to score, and must return a named numeric with scores. The higher the score, the more important the feature. The calling function (`calculate()`) ensures that the returned vector gets sorted and that missing feature scores get a score value of `NA`.

Partial Scoring

Some features support partial scoring of the feature set: If `nfeat` is not `NULL`, only the best `nfeat` features are guaranteed to get a score. Additional features may be ignored for computational reasons, and then get a score value of `NA`.

See Also

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `mlr_filters`

`FilterAnova` *ANOVA F-Test Filter*

Description

ANOVA F-Test filter calling `stats::aov()`. Note that this is equivalent to a t-test for binary classification.

The filter value is `-log10(p)` where `p` is the p-value. This transformation is necessary to ensure numerical stability for very small p-values.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterAnova$new()
mlr_filters$get("anova")
flt("anova")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("iris")
filter = flt("anova")
filter$calculate(task)
head(as.data.table(filter), 3)

# transform to p-value
10^(-filter$scores)
```

FilterAUC *AUC Filter*

Description

Area under the (ROC) Curve filter, analogously to `mlr3measures::auc()` from **mlr3measures**. Missing values of the features are removed before calculating the AUC. If the AUC is undefined for the input, it is set to 0.5 (random classifier). The absolute value of the difference between the AUC and 0.5 is used as final filter value.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterAUC$new()
mlr_filters$get("auc")
flt("auc")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAnova`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("pima")
filter = flt("auc")
filter$calculate(task)
head(as.data.table(filter), 3)
```

FilterCarScore *Conditional Mutual Information Based Feature Selection Filter*

Description

Calculates the Correlation-Adjusted (marginal) coRelation scores (short CAR scores) implemented in `care::carscore()` in package **care**. The CAR scores for a set of features are defined as the correlations between the target and the decorrelated features. The filter returns the absolute value of the calculated scores.

Argument `verbose` defaults to `FALSE`.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterCarScore$new()
mlr_filters$get("carscore")
flt("carscore")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCMIM`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("mtcars")
filter = flt("carscore")
filter$calculate(task)
head(as.data.table(filter), 3)

## changing filter settings
filter = flt("carscore")
filter$param_set$values = list("diagonal" = TRUE)
filter$calculate(task)
head(as.data.table(filter), 3)
```

FilterCMIM *Minimal Conditional Mutual Information Filter*

Description

Minimal conditional mutual information maximisation filter calling `praznik::CMIM()` from package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterCMIM$new()
mlr_filters$get("cmim")
flt("cmim")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("iris")
filter = flt("cmim")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterCorrelation *Correlation Filter*

Description

Simple correlation filter calling `stats::cor()`. The filter score is the absolute value of the correlation.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterCorrelation$new()
mlr_filters$get("correlation")
flt("correlation")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterDISR, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, FilterVariance, Filter, mlr_filters

Examples

```
## Pearson (default)
task = mlr3::tsk("mtcars")
filter = flt("correlation")
filter$calculate(task)
as.data.table(filter)

## Spearman
filter = FilterCorrelation$new()
filter$param_set$values = list("method" = "spearman")
filter$calculate(task)
as.data.table(filter)
```

FilterDISR *Double Input Symmetrical Relevance Filter*

Description

Double input symmetrical relevance filter calling praznik::DISR() from package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterDISR$new()
mlr_filters$get("disc")
flt("disc")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, FilterVariance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("iris")
filter = flt("disr")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterImportance *Filter for Embedded Feature Selection via Variable Importance*

Description

Variable Importance filter using embedded feature selection of machine learning algorithms. Takes a mlr3::Learner which is capable of extracting the variable importance (property "importance"), fits the model and extracts the importance values to use as filter scores.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterImportance$new(learner = mlr3::lrn("classif.rpart"))
mlr_filters$get("importance")
flt("importance")
```

- `learner` :: mlr3::Learner
  Learner to extract the importance values from.

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, FilterVariance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("iris")
learner = mlr3::lrn("classif.rpart")
filter = flt("importance", learner = learner)
filter$calculate(task)
as.data.table(filter)
```

FilterInformationGain *Information Gain Filter*

Description

Information gain filter calling `FSelectorRcpp::information_gain()` in package **FSelectorRcpp**. Set parameter `"type"` to `"gainratio"` to calculate the gain ratio, or set to `"symuncert"` to calculate the symmetrical uncertainty (see `FSelectorRcpp::information_gain()`). Default is `"infogain"`.

Argument `equal` defaults to `FALSE` for classification tasks, and to `TRUE` for regression tasks.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterInformationGain$new()
mlr_filters$get("information_gain")
flt("information_gain")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
## InfoGain (default)
task = mlr3::tsk("pima")
filter = flt("information_gain")
filter$calculate(task)
head(filter$scores, 3)
as.data.table(filter)

## GainRatio

filterGR = flt("information_gain")
filterGR$param_set$values = list("type" = "gainratio")
filterGR$calculate(task)
head(as.data.table(filterGR), 3)
```

FilterJMI *Joint Mutual Information Filter*

Description

Joint mutual information filter calling `praznik::JMI()` in package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterJMI$new()
mlr_filters$get("jmi")
flt("jmi")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterKruskalTest`, `FilterMIM`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("iris")
filter = flt("jmi")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterJMIM *Minimal Joint Mutual Information Maximisation Filter*

Description

Minimal joint mutual information maximisation filter calling `praznik::JMIM()` in package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterJMIM$new()
mlr_filters$get("jmim")
flt("jmim")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterImportance, FilterInformationGain, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, FilterVariance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("iris")
filter = flt("jmim")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterKruskalTest *Kruskal-Wallis Test Filter*

Description

Kruskal-Wallis rank sum test filter calling `stats::kruskal.test()`.

The filter value is `-log10(p)` where `p` is the p-value. This transformation is necessary to ensure numerical stability for very small p-values.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterKruskalTest$new()
mlr_filters$get("kruskal_test")
flt("kruskal_test")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, FilterVariance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("iris")
filter = flt("kruskal_test")
filter$calculate(task)
as.data.table(filter)

# transform to p-value
10^(-filter$scores)
```

`FilterMIM` *Conditional Mutual Information Based Feature Selection Filter*

Description

Conditional mutual information based feature selection filter calling `praznik::MIM()` in package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterMIM$new()
mlr_filters$get("mim")
flt("mim")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMRMR`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("iris")
filter = flt("mim")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterMRMR *Minimum redundancy maximal relevancy filter*

Description

Minimum redundancy maximal relevancy filter calling `praznik::MRMR()` in package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterMRMR$new()
mlr_filters$get("mrmr")
flt("mrmr")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: `FilterAUC`, `FilterAnova`, `FilterCMIM`, `FilterCarScore`, `FilterCorrelation`, `FilterDISR`, `FilterImportance`, `FilterInformationGain`, `FilterJMIM`, `FilterJMI`, `FilterKruskalTest`, `FilterMIM`, `FilterNJMIM`, `FilterPerformance`, `FilterVariance`, `Filter`, `mlr_filters`

Examples

```
task = mlr3::tsk("iris")
filter = flt("mrmr")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterNJMIM *Minimal Normalised Joint Mutual Information Maximisation Filter*

Description

Minimal normalised joint mutual information maximisation filter calling `praznik::NJMIM()` from package **praznik**.

This filter supports partial scoring (see Filter).

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterNJMIM$new()
mlr_filters$get("njmim")
flt("njmim")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterPerformance, FilterVariance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("iris")
filter = flt("njmim")
filter$calculate(task, nfeat = 2)
as.data.table(filter)
```

FilterPerformance *Predictive Performance Filter*

Description

Filter which uses the predictive performance of a mlr3::Learner as filter score. Performs a `mlr3::resample()` for each feature separately. The filter score is the aggregated performance of the mlr3::Measure, or the negated aggregated performance if the measure has to be minimized.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterPerformance$new(learner = mlr3::lrn("classif.rpart"),
  resampling = mlr3::rsmp("holdout"), measure = mlr3::msr("classif.ce"))
mlr_filters$get("performance")
flt("performance")
```

- `learner` :: mlr3::Learner.
- `resampling` :: mlr3::Resampling.
- `measure` :: mlr3::Measure.

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterVariance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("iris")
learner = mlr3::lrn("classif.rpart")
filter = flt("performance", learner = learner)
filter$calculate(task)
as.data.table(filter)
```

FilterVariance *Variance Filter*

Description

Variance filter calling `stats::var()`.

Argument `na.rm` defaults to `TRUE` here.

Format

R6::R6Class inheriting from Filter.

Construction

```
FilterVariance$new()
mlr_filters$get("variance")
flt("variance")
```

See Also

Dictionary of Filters: mlr_filters

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, Filter, mlr_filters

Examples

```
task = mlr3::tsk("mtcars")
filter = flt("variance")
filter$calculate(task)
head(filter$scores, 3)
as.data.table(filter)
```

flt *Syntactic Sugar for Filter Construction*

Description

This function complements mlr_filters with a function in the spirit of mlr3::mlr_sugar.

Usage

```
flt(.key, ...)
```

Arguments

`.key`	:: `character(1)` Key passed to the respective mlr3misc::Dictionary to retrieve the object.
`...`	:: named `list()` Named arguments passed to the constructor, to be set as parameters in the paradox::ParamSet, or to be set as public field. See `mlr3misc::dictionary_sugar_get()` for more details.

Value

Filter.

Examples

```
flt("correlation", method = "kendall")
```

mlr_filters *Dictionary of Filters*

Description

A simple Dictionary storing objects of class Filter. Each Filter has an associated help page, see `mlr_filters_[id]`.

This dictionary can get populated with additional filters by add-on packages.

For a more convenient way to retrieve and construct filters, see `flt()`.

Usage

```
mlr_filters
```

Format

R6Class object

Usage

See Dictionary.

See Also

Other Filter: FilterAUC, FilterAnova, FilterCMIM, FilterCarScore, FilterCorrelation, FilterDISR, FilterImportance, FilterInformationGain, FilterJMIM, FilterJMI, FilterKruskalTest, FilterMIM, FilterMRMR, FilterNJMIM, FilterPerformance, FilterVariance, Filter

Examples

```
mlr_filters$keys()
as.data.table(mlr_filters)
mlr_filters$get("mim")
flt("anova")
```

Index

∗Topic **datasets**
    mlr_filters, 17

care::carscore(), *6*

Dictionary, *4–18*
dictionary, *3*

Filter, 3, *4–18*
FilterAnova, *4*, 4, *5–16*, *18*
FilterAUC, *4*, 5, *6–16*, *18*
FilterCarScore, *4, 5*, 6, *7–16*, *18*
FilterCMIM, *4–6*, 7, *8–16*, *18*
FilterCorrelation, *4–7*, 7, *9–16*, *18*
FilterDISR, *4–8*, 8, *9–16*, *18*
FilterImportance, *4–9*, 9, *10–16*, *18*
FilterInformationGain, *4–9*, 10, *11–16*, *18*
FilterJMI, *4–10*, 11, *12–16*, *18*
FilterJMIM, *4–11*, 11, *12–16*, *18*
FilterKruskalTest, *4–12*, 12, *13–16*, *18*
FilterMIM, *4–12*, 13, *14–16*, *18*
FilterMRMR, *4–13*, 14, *15, 16*, *18*
FilterNJMIM, *4–14*, 14, *16*, *18*
FilterPerformance, *4–15*, 15, *16*, *18*
Filters, *4–16*
FilterVariance, *4–16*, 16, *18*
flt, 17
flt(), *17*
FSelectorRcpp::information_gain(), *10*

mlr3::Learner, *9*, *15*
mlr3::Measure, *15*
mlr3::mlr_sugar, *17*
mlr3::resample(), *15*
mlr3::Resampling, *15*
mlr3::Task, *3, 4*
mlr3filters *(*mlr3filters-package*)*, 2
mlr3filters-package, 2
mlr3measures::auc(), *5*
mlr3misc::Dictionary, *17*
mlr3misc::dictionary_sugar_get(), *17*
mlr_filters, *3–17*, 17
mlr_filters_anova *(*FilterAnova*)*, 4
mlr_filters_auc *(*FilterAUC*)*, 5
mlr_filters_carscore *(*FilterCarScore*)*, 6
mlr_filters_cmim *(*FilterCMIM*)*, 7
mlr_filters_correlation
        *(*FilterCorrelation*)*, 7
mlr_filters_disr *(*FilterDISR*)*, 8
mlr_filters_information_gain
        *(*FilterInformationGain*)*, 10
mlr_filters_jmi *(*FilterJMI*)*, 11
mlr_filters_jmim *(*FilterJMIM*)*, 11
mlr_filters_kruskal_test
        *(*FilterKruskalTest*)*, 12
mlr_filters_mim *(*FilterMIM*)*, 13
mlr_filters_mrmr *(*FilterMRMR*)*, 14
mlr_filters_njmim *(*FilterNJMIM*)*, 14
mlr_filters_performance
        *(*FilterPerformance*)*, 15
mlr_filters_variable_importance
        *(*FilterImportance*)*, 9
mlr_filters_variance *(*FilterVariance*)*,
        16
mlr_reflections$task_feature_types, *3*
mlr_reflections$task_properties, *3*

paradox::ParamSet, *3*, *17*
praznik::CMIM(), *7*
praznik::DISR(), *8*
praznik::JMI(), *11*
praznik::JMIM(), *11*
praznik::MIM(), *13*
praznik::MRMR(), *14*
praznik::NJMIM(), *14*

R6::R6Class, *3–16*
R6Class, *17*
requireNamespace(), *3*

stats::aov(), *4*
stats::cor(), *7*
stats::kruskal.test(), *12*
stats::var(), *16*