Package ‘mlr3cluster’

April 6, 2022

Title Cluster Extension for ‘mlr3’
Version 0.1.3
Description Extends the ‘mlr3’ package with cluster analysis.
License LGPL-3

URL https://mlr3cluster.mlr-org.com,
https://github.com/mlr-org/mlr3cluster

BugReports https://github.com/mlr-org/mlr3cluster/issues
Depends R (>= 3.1.0), mlr3 (>= 0.13.3)
Imports backports (>= 1.1.10), checkmate, clue, clusterCrit,
data.table, mlr3misc (>= 0.9.4), paradox, R6, stats
Suggests dbscan, e1071, ClusterR, kernlab, apcluster, LPCM, mlbench,
RWeka, testthat

Encoding UTF-8
RoxygenNote 7.1.2

Collate ‘LearnerClust.R’ ‘LearnerClustAffinityPropagation.R’
‘LearnerClustAgnes.R’ ‘LearnerClustCMeans.R’
‘LearnerClustCobweb.R’ ‘LearnerClustDBSCAN.R’
‘LearnerClustDiana.R’ ‘LearnerClustEM.R’ ‘LearnerClustFanny.R’
‘LearnerClustFarthestFirst.R’ ‘LearnerClustFeatureless.R’
‘LearnerClustHclust.R’ ‘LearnerClustKKMeans.R’
‘LearnerClustKMeans.R’ ‘LearnerClustMeanShift.R’
‘LearnerClustMiniBatchKMeans.R’ ‘LearnerClustPAM.R’
‘LearnerClustSimpleKMeans.R’ ‘LearnerClustXMeans.R’
‘TaskClust_usarrest.R’ ‘as_prediction_clust.R’
‘as_task_clust.R’ ‘helper.R’ ‘zzz.R’

NeedsCompilation no

Author Damir Pulatov [cre, aut],
   Michel Lang [aut] (<https://orcid.org/0000-0001-9754-0393>)

Maintainer Damir Pulatov <damirpolat@protonmail.com>
Description

Extends the `mlr3` package with cluster analysis.
as_prediction_clust

Author(s)

Maintainer: Damir Pulatov <damirpolat@protonmail.com>

Authors:

- Michel Lang <michellang@gmail.com> (ORCID)

See Also

Useful links:

- https://mlr3cluster.mlr-org.com
- https://github.com/mlr-org/mlr3cluster
- Report bugs at https://github.com/mlr-org/mlr3cluster/issues

as_prediction_clust       Convert to a Cluster Prediction

Description

Convert object to a PredictionClust.

Usage

as_prediction_clust(x, ...)

## S3 method for class 'PredictionClust'
as_prediction_clust(x, ...)

## S3 method for class 'data.frame'
as_prediction_clust(x, ...)

Arguments

x (any)
Object to convert.

... (any)
Additional arguments.

Value

PredictionClust.
Examples

```r
# create a prediction object
task = tsk("usarrests")
learner = lrn("clust.kmeans")
learner = lrn("clust.cmeans", predict_type = "prob")
learner$train(task)
p = learner$predict(task)

# convert to a data.table
tab = as.data.table(p)

# convert back to a Prediction
as_prediction_clust(tab)

# split data.table into a 3 data.tables based on UrbanPop
f = cut(task$data(rows = tab$row_ids)$UrbanPop, 3)
tabs = split(tab, f)

# convert back to list of predictions
preds = lapply(tabs, as_prediction_clust)

# calculate performance in each group
sapply(preds, function(p) p$score(task = task))
```

as_task_clust  

Convert to a Cluster Task

Description

Convert object to a TaskClust. This is a S3 generic, specialized for at least the following objects:

1. TaskClust: ensure the identity.
2. `data.frame()` and DataBackend: provides an alternative to calling constructor of TaskClust.

Usage

```r
as_task_clust(x, ...)
```

## S3 method for class 'TaskClust'
as_task_clust(x, clone = FALSE, ...)

## S3 method for class 'data.frame'
as_task_clust(x, id = deparse(substitute(x)), ...)

## S3 method for class 'DataBackend'
as_task_clust(x, id = deparse(substitute(x)), ...)

## S3 method for class 'formula'
as_task_clust(x, data, id = deparse(substitute(data)), ...)

```
Arguments

- **x**: (any) Object to convert.
- **...**: (any) Additional arguments.
- **clone**: (logical(1)) If TRUE, ensures that the returned object is not the same as the input x.
- **id**: (character(1)) Id for the new task. Defaults to the (deparsed and substituted) name of the data argument.
- **data**: (data.frame()) Data frame containing all columns specified in formula x.

Value

```
TaskClust.
```

Examples

```
as_task_clust(datasets::USArrests)
```

---

LearnerClust  
*Cluster Learner*

Description

This Learner specializes `mlr3::Learner` for cluster problems:

- task_type is set to "clust".
- Creates Predictions of class `PredictionClust`.
- Possible values for `predict_types` are:
  - "partition": Integer indicating the cluster membership.
  - "prob": Probability for belonging to each cluster.

Predefined learners can be found in the `mlr3misc::Dictionary mlr3::mlr_learners`.

Super class

```
mlr3::Learner -> LearnerClust
```

Public fields

- **assignments**: (NULL | vector()) Cluster assignments from learned model.
- **save_assignments**: (logical()) Should assignments for 'train' data be saved in the learner? Default is TRUE.
Methods

Public methods:

• LearnerClust$new()
• LearnerClust$reset()
• LearnerClust$clone()

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

Usage:
LearnerClust$new(
  id,
  param_set = ps(),
  predict_types = "partition",
  feature_types = character(),
  properties = character(),
  packages = character(),
  label = NA_character_
)

Arguments:

id (character(1))
  Identifier for the new instance.
param_set (paradox::ParamSet)
  Set of hyperparameters.
predict_types (character())
  Supported predict types. Must be a subset of mlr_reflections$learner_predict_types.
feature_types (character())
  Feature types the learner operates on. Must be a subset of mlr_reflections$task_feature_types.
properties (character())
  Set of properties of the Learner. Must be a subset of mlr_reflections$learner_properties.
The following properties are currently standardized and understood by learners in mlr3:
  • "missings": The learner can handle missing values in the data.
  • "weights": The learner supports observation weights.
  • "importance": The learner supports extraction of importance scores, i.e. comes with an $importance() extractor function (see section on optional extractors in Learner).
  • "selected_features": The learner supports extraction of the set of selected features, i.e. comes with a $selected_features() extractor function (see section on optional extractors in Learner).
  • "oob_error": The learner supports extraction of estimated out of bag error, i.e. comes with a oob_error() extractor function (see section on optional extractors in Learner).
packages (character())
  Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().
label (character(1))
  Label for the new instance.

Method reset(): Reset assignments field before calling parent’s reset().
Usage:
LearnerClust$reset()

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

Usage:
LearnerClust$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

Examples

library(mlr3)
library(mlr3cluster)
ids = mlr_learners$keys("clust")
ids

# get a specific learner from mlr_learners:
lrn = mlr_learners$get("clust.kmeans")
print(lrn)

MeasureClust        Cluster Measure

Description

This measure specializes mlr3::Measure for cluster analysis:

- task_type is set to "clust".
- Possible values for predict_type are "partition" and "prob".

Predefined measures can be found in the mlr3misc::Dictionary mlr3::mlr_measures.

Super class

mlr3::Measure -> MeasureClust

Methods

Public methods:

- MeasureClust$new()

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

Usage:
MeasureClust$new(
  id,
  range,
  minimize = NA,
  aggregator = NULL,
  properties = character(),
  predict_type = "partition",
  task_properties = character(),
  packages = character(),
  label = NA_character_,
  man = NA_character_
)

Arguments:
id (character(1))
  Identifier for the new instance.
range (numeric(2))
  Feasible range for this measure as c(lower_bound,upper_bound). Both bounds may be infinite.
minimize (logical(1))
  Set to TRUE if good predictions correspond to small values, and to FALSE if good predictions correspond to large values. If set to NA (default), tuning this measure is not possible.
aggregator (function(x))
  Function to aggregate individual performance scores x where x is a numeric vector. If NULL, defaults to mean().
properties (character())
  Properties of the measure. Must be a subset of mlr_reflections$measure_properties. Supported by mlr3:
  • "requires_task" (requires the complete Task),
  • "requires_learner" (requires the trained Learner),
  • "requires_train_set" (requires the training indices from the Resampling), and
  • "na_score" (the measure is expected to occasionally return NA or NaN).
predict_type (character(1))
  Required predict type of the Learner. Possible values are stored in mlr_reflections$learner_predict_types.
task_properties (character())
  Required task properties, see Task.
packages (character())
  Set of required packages. A warning is signaled by the constructor if at least one of the packages is not installed, but loaded (not attached) later on-demand via requireNamespace().
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().

See Also

Example cluster measures: clust.dunn
Description

A LearnerClust for agglomerative hierarchical clustering implemented in \texttt{cluster::agnes()}. The predict method uses \texttt{stats::cutree()} which cuts the tree resulting from hierarchical clustering into specified number of groups (see parameter \texttt{k}). The default number for \texttt{k} is 2.

Dictionary

This Learner can be instantiated via the dictionary \texttt{mlr_learners} or with the associated sugar function \texttt{lrn()}:

\begin{verbatim}
mlr_learners$get("clust.agnes")
lrn("clust.agnes")
\end{verbatim}

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: \texttt{mlr}, \texttt{mlr3cluster}, \texttt{cluster}

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>character</td>
<td>euclidean</td>
<td>euclidean, manhattan</td>
<td>-</td>
</tr>
<tr>
<td>stand</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>method</td>
<td>character</td>
<td>average</td>
<td>average, single, complete, ward, weighted, flexible, gaverage</td>
<td>-</td>
</tr>
<tr>
<td>trace.lev</td>
<td>integer</td>
<td>0</td>
<td></td>
<td>[0, \infty)</td>
</tr>
<tr>
<td>k</td>
<td>integer</td>
<td>2</td>
<td></td>
<td>[1, \infty)</td>
</tr>
<tr>
<td>par.method</td>
<td>untyped</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

Super classes

\texttt{mlr3::Learner} \rightarrow \texttt{mlr3cluster::LearnerClust} \rightarrow LearnerClustAgnes

Methods

Public methods:
- **LearnerClustAgnes$new()**
- **LearnerClustAgnes$clone()**

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

*Usage:*

LearnerClustAgnes$new()

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

*Usage:*

LearnerClustAgnes$clone(deep = FALSE)

*Arguments:*

depth Whether to make a deep clone.

**Examples**

```r
learner = mlr3::lrn("clust.agnes")
print(learner)

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

---

**mlr_learners_clust.ap  Affinity Propagation Clustering Learner**

**Description**

A **LearnerClust** for Affinity Propagation clustering implemented in `apcluster::apcluster()`. `apcluster::apcluster()` doesn’t have set a default for similarity function. Therefore, the s parameter here is set to `apcluster::negDistMat(r = 2L)` by default since this is what is used in the original paper on Affity Propagation clustering. The predict method computes the closest cluster exemplar to find the cluster memberships for new data. The code is taken from StackOverflow answer by the `apcluster` package maintainer.

**Dictionary**

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

```r
mlr_learners$get("clust.ap")
lrn("clust.ap")
```

**Meta Information**

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3, mlr3cluster, apcluster**
Parameters
<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>untyped</td>
<td>apcluster::negDistMat, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>untyped</td>
<td>NA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>numeric</td>
<td>-</td>
<td></td>
<td>[0, 1]</td>
</tr>
<tr>
<td>maxits</td>
<td>integer</td>
<td>1000</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>convits</td>
<td>integer</td>
<td>100</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>lam</td>
<td>numeric</td>
<td>0.9</td>
<td></td>
<td>[0.5, 1]</td>
</tr>
<tr>
<td>includeSim</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>details</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>nonoise</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>seed</td>
<td>integer</td>
<td>-</td>
<td></td>
<td>(−∞, ∞)</td>
</tr>
</tbody>
</table>

**Super classes**

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustAP`

**Methods**

**Public methods:**

- `LearnerClustAP$new()`
- `LearnerClustAP$clone()`

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

*Usage:*

```r
LearnerClustAP$new()
```

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

*Usage:*

```r
LearnerClustAP$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.

**Examples**

```r
learner = mlr3::lrn("clust.ap")
print(learner)

# available parameters:
learner$param_set$ids()
```
**Description**

A LearnerClust for fuzzy clustering implemented in `e1071::cmeans()`. `e1071::cmeans()` doesn’t have a default value for the number of clusters. Therefore, the centers parameter here is set to 2 by default. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

**Dictionary**

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

```r
mlr_learners$get("clust.cmeans")
lrn("clust.cmeans")
```

**Meta Information**

- Task type: “clust”
- Predict Types: “partition”, “prob”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3`, `mlr3cluster`, `e1071`

**Parameters**

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>centers</td>
<td>untyped</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>iter.max</td>
<td>integer</td>
<td>100</td>
<td>TRUE, FALSE</td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>FALSE</td>
<td>FALSE</td>
<td>-</td>
</tr>
<tr>
<td>dist</td>
<td>character</td>
<td>euclidean</td>
<td>euclidean, manhattan</td>
<td>-</td>
</tr>
<tr>
<td>method</td>
<td>character</td>
<td>cmeans</td>
<td>cmeans, ufcl</td>
<td>-</td>
</tr>
<tr>
<td>m</td>
<td>numeric</td>
<td>2</td>
<td>[1, ∞)</td>
<td>-</td>
</tr>
<tr>
<td>rate.par</td>
<td>numeric</td>
<td>-</td>
<td>[0, 1]</td>
<td>-</td>
</tr>
<tr>
<td>weights</td>
<td>untyped</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>control</td>
<td>untyped</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Super classes**

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustCMeans`
Methods

Public methods:

• LearnerClustCMeans$new()
• LearnerClustCMeans$clone()

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

Method clone(): The objects of this class are cloneable with this method.
Usage:
LearnerClustCMeans$clone(deep = FALSE)
Arguments:
deep  Whether to make a deep clone.

Examples

learner = mlr3::lrn("clust.cmeans")
print(learner)

# available parameters:
learner$param_set$ids()
Parameters
### Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustCobweb`

### Methods

**Public methods:**

- `LearnerClustCobweb$new()`
- `LearnerClustCobweb$clone()`

#### Method `new()`:

Creates a new instance of this R6 class.

**Usage:**

```r
LearnerClustCobweb$new()
```

#### Method `clone()`:

The objects of this class are cloneable with this method.

**Usage:**

```r
LearnerClustCobweb$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.

### Examples

```r
learner = mlr3::lrn("clust.cobweb")
print(learner)

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

---

### Description

A LearnerClust for density-based clustering implemented in `dbscan::dbscan()`. The predict method uses `dbscan::predict.dbscan_fast()` to compute the cluster memberships for new data.
Dictionary

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

```r
mlr_learners$get("clust.dbscan")
lrn("clust.dbscan")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: **mlr3, mlr3cluster, dbscan**

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>eps</td>
<td>numeric</td>
<td>-</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>minPts</td>
<td>integer</td>
<td>5</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>borderPoints</td>
<td>logical</td>
<td>TRUE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>weights</td>
<td>untyped</td>
<td>-</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>search</td>
<td>character</td>
<td>kdtree</td>
<td>kdtree, linear, dist</td>
<td>-</td>
</tr>
<tr>
<td>bucketSize</td>
<td>integer</td>
<td>10</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>splitRule</td>
<td>character</td>
<td>SUGGEST</td>
<td>STD, MIDPT, FAIR, SL_MIDPT, SL_FAIR, SUGGEST</td>
<td>-</td>
</tr>
<tr>
<td>approx</td>
<td>numeric</td>
<td>0</td>
<td></td>
<td>(−∞, ∞)</td>
</tr>
</tbody>
</table>

Super classes

**mlr3::Learner** -**mlr3cluster::LearnerClust** -**LearnerClustDBSCAN**

Methods

**Public methods:**
- `LearnerClustDBSCAN$new()`
- `LearnerClustDBSCAN$clone()`

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

**Usage:**
```
LearnerClustDBSCAN$new()
```

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

**Usage:**
```
LearnerClustDBSCAN$clone(deep = FALSE)
```
**Arguments:**

- `deep` Whether to make a deep clone.

**Examples**

```r
learner = mlr3::lrn("clust.dbscan")
print(learner)

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

---

### mlr_learners_clust.diana

**Divisive Hierarchical Clustering Learner**

---

**Description**

A LearnerClust for divisive hierarchical clustering implemented in `cluster::diana()`. The predict method uses `stats::cutree()` which cuts the tree resulting from hierarchical clustering into specified number of groups (see parameter `k`). The default value for `k` is 2.

**Dictionary**

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

```r
dlrs_learners$get("clust.diana")
lrn("clust.diana")
```

**Meta Information**

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3`, `mlr3cluster`, `cluster`

**Parameters**

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>character</td>
<td>euclidean</td>
<td>euclidean, manhattan</td>
<td>-</td>
</tr>
<tr>
<td>stand</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>trace.lev</td>
<td>integer</td>
<td>0</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>k</td>
<td>integer</td>
<td>2</td>
<td></td>
<td>[1, ∞)</td>
</tr>
</tbody>
</table>
Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustDiana

Methods

Public methods:

• LearnerClustDiana$new()
• LearnerClustDiana$clone()

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

Usage:
LearnerClustDiana$new()

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

Usage:
LearnerClustDiana$clone(deep = FALSE)

Arguments:
  deep  Whether to make a deep clone.

Examples

learner = mlr3::lrn("clust.diana")
print(learner)

# available parameters:
learner$param_set$ids()

mlr_learners_clust.em  Expectation-Maximization Clustering Learner

Description

A LearnerClust for Expectation-Maximization clustering implemented in RWeka::list_Weka_interfaces().
The predict method uses RWeka::predict.Weka_clusterer() to compute the cluster memberships for new data.

Dictionary

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

mlr_learners$get("clust.em")
lrn("clust.em")
Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: mlr3, mlr3cluster, RWeka

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>integer</td>
<td>100</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>ll_cv</td>
<td>numeric</td>
<td>1e-06</td>
<td></td>
<td>[1e−06, ∞)</td>
</tr>
<tr>
<td>ll_iter</td>
<td>numeric</td>
<td>1e-06</td>
<td></td>
<td>[1e−06, ∞)</td>
</tr>
<tr>
<td>M</td>
<td>numeric</td>
<td>1e-06</td>
<td></td>
<td>[1e−06, ∞)</td>
</tr>
<tr>
<td>max</td>
<td>integer</td>
<td>-1</td>
<td></td>
<td>[-1, ∞)</td>
</tr>
<tr>
<td>N</td>
<td>integer</td>
<td>-1</td>
<td></td>
<td>[-1, ∞)</td>
</tr>
<tr>
<td>num_slots</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>S</td>
<td>integer</td>
<td>100</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>X</td>
<td>integer</td>
<td>10</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>K</td>
<td>integer</td>
<td>10</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>V</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>output_debug_info</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
</tbody>
</table>

Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustEM

Methods

Public methods:

- LearnerClustEM$new()
- LearnerClustEM$clone()

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

Usage:
LearnerClustEM$new()

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

Usage:
LearnerClustEM$clone(deep = FALSE)

Arguments:

- deep Whether to make a deep clone.
Examples

```r
learner = mlr3::lrn("clust.em")
print(learner)

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

---

**Fuzzy Analysis Clustering Learner**

**Description**

A LearnerClust for fuzzy clustering implemented in `cluster::fanny()`. `cluster::fanny()` doesn’t have a default value for the number of clusters. Therefore, the `k` parameter which corresponds to the number of clusters here is set to 2 by default. The predict method copies cluster assignments and memberships generated for train data. The predict does not work for new data.

**Dictionary**

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

```r
mlr_learners$get("clust.fanny")
lrn("clust.fanny")
```

**Meta Information**

- Task type: “clust”
- Predict Types: “partition”, “prob”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3`, `mlr3cluster`, `cluster`

**Parameters**

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>integer</td>
<td>2</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>memb.exp</td>
<td>numeric</td>
<td>2</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>metric</td>
<td>character</td>
<td></td>
<td>euclidean, manhattan, SqEuclidean</td>
<td>-</td>
</tr>
<tr>
<td>stand</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>maxit</td>
<td>integer</td>
<td>500</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>tol</td>
<td>numeric</td>
<td>1e-15</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>trace.lev</td>
<td>integer</td>
<td>0</td>
<td></td>
<td>[0, ∞)</td>
</tr>
</tbody>
</table>
Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustFanny

Methods

Public methods:

- LearnerClustFanny$new()
- LearnerClustFanny$clone()

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

Usage:
LearnerClustFanny$new()

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

Usage:
LearnerClustFanny$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Examples

learner = mlr3::lrn("clust.fanny")
print(learner)

# available parameters:
learner$param_set$ids()
Meta Information

- Task type: “clust”
- Predict Types: “partition”, “prob”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3, mlr3cluster`

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>num_clusters</td>
<td>integer</td>
<td>1</td>
<td>[1, ∞)</td>
</tr>
</tbody>
</table>

Super classes

`mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustFeatureless`

Methods

**Public methods:**
- `LearnerClustFeatureless$new()`
- `LearnerClustFeatureless$clone()`

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

*Usage:*

```
LearnerClustFeatureless$new()
```

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

*Usage:*

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

*Arguments:*

depth: Whether to make a deep clone.

Examples

```r
learner = mlr3::lrn("clust.featureless")
print(learner)

# available parameters:
learner$param_set$ids()
```
**mlr_learners_clust.FF**  
*Farthest First Clustering Learner*

**Description**

A LearnerClust for Farthest First clustering implemented in R\texttt{Weka::FarthestFirst()}. The predict method uses R\texttt{Weka::predict.Weka_clusterer()} to compute the cluster memberships for new data.

**Dictionary**

This Learner can be instantiated via the dictionary mlr_learners or with the associated sugar function \texttt{lrn()}:

\begin{verbatim}
mlr_learners$get("clust.ff")
lrn("clust.ff")
\end{verbatim}

**Meta Information**

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: \texttt{mlr3}, \texttt{mlr3cluster}, \texttt{RWeka}

**Parameters**

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>integer</td>
<td>2</td>
<td></td>
<td>[1, (\infty)]</td>
</tr>
<tr>
<td>S</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>[1, (\infty)]</td>
</tr>
<tr>
<td>output_debug_info</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
</tbody>
</table>

**Super classes**

\texttt{mlr3::Learner} \rightarrow \texttt{mlr3cluster::LearnerClust} \rightarrow LearnerClustFF

**Methods**

- **Public methods:**
  - \texttt{LearnerClustFarthestFirst$new()}
  - \texttt{LearnerClustFarthestFirst$clone()}

**Method new():** Creates a new instance of this \texttt{R6} class.
Usage:
LearnerClustFarthestFirst$new()

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

Usage:
LearnerClustFarthestFirst$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

Examples
learner = mlr3::lrn("clust.ff")
print(learner)

# available parameters:
learner$param_set$ids()

mlr_learners_clust.hclust

Agglomerative Hierarchical Clustering Learner

Description
A LearnerClust for agglomerative hierarchical clustering implemented in stats::hclust(). Difference Calculation is done by stats::dist()

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

mlr_learners$get("clust.hclust")
lrn("clust.hclust")

Meta Information
• Task type: “clust”
• Predict Types: “partition”
• Feature Types: “logical”, “integer”, “numeric”
• Required Packages: mlr3, mlr3cluster, 'stats'
### Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>character</td>
<td>complete</td>
<td>ward.D, ward.D2, single, complete, average, mcquitty, median, centroid</td>
<td>-</td>
</tr>
<tr>
<td>members</td>
<td>untyped</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>distmethod</td>
<td>character</td>
<td>euclidean</td>
<td>euclidean, maximum, manhattan, canberra, binary, minkowski</td>
<td>-</td>
</tr>
<tr>
<td>diag</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>upper</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>numeric</td>
<td>2</td>
<td></td>
<td>$(-\infty, \infty)$</td>
</tr>
<tr>
<td>k</td>
<td>integer</td>
<td>2</td>
<td></td>
<td>$[1, \infty)$</td>
</tr>
</tbody>
</table>

### Super classes

\texttt{mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustHclust}

### Methods

**Public methods:**

- \texttt{LearnerClustHclust$new()}
- \texttt{LearnerClustHclust$clone()}

**Method \texttt{new()}:** Creates a new instance of this \texttt{R6} class.

*Usage:*

\texttt{LearnerClustHclust$new()}

**Method \texttt{clone()}:** The objects of this class are cloneable with this method.

*Usage:*

\texttt{LearnerClustHclust$clone(deep = FALSE)}

*Arguments:*

- \texttt{deep} Whether to make a deep clone.

### Examples

```r
learner = mlr3::lrn("clust.hclust")
print(learner)

# available parameters:
learner$param_set$ids()
```
Kernel K-Means Clustering Learner

Description

A LearnerClust for kernel k-means clustering implemented in `kernlab::kkmeans()`. `kernlab::kkmeans()` doesn't have a default value for the number of clusters. Therefore, the `centers` parameter here is set to 2 by default. Kernel parameters have to be passed directly and not by using the `kpar` list in `kkmeans`. The predict method finds the nearest center in kernel distance to assign clusters for new data points.

Dictionary

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

```r
mlr_learners$get("clust.kkmeans")
lrn("clust.kkmeans")
```

Meta Information

- Task type: “clust"
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3`, `mlr3cluster`, `kernlab`

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>centers</td>
<td>untyped</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>kernel</td>
<td>character</td>
<td>rbfdot</td>
<td>vanilladot, polydot, rbfdot, tanhdot, laplacedot, besseldot, anovadot, splinedot</td>
<td></td>
</tr>
<tr>
<td>sigma</td>
<td>numeric</td>
<td>-</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>degree</td>
<td>integer</td>
<td>3</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>1</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>offset</td>
<td>numeric</td>
<td>1</td>
<td></td>
<td>(-∞, ∞)</td>
</tr>
<tr>
<td>order</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>(-∞, ∞)</td>
</tr>
<tr>
<td>alg</td>
<td>character</td>
<td>kkmeans</td>
<td>kkmeans, kerninghan</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>numeric</td>
<td>1</td>
<td></td>
<td>(-∞, ∞)</td>
</tr>
</tbody>
</table>
Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustKKMeans`

Methods

**Public methods:**
- `LearnerClustKKMeans$new()`
- `LearnerClustKKMeans$clone()`

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

*Usage:*

```r
LearnerClustKKMeans$new()
```

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

*Usage:*

```r
LearnerClustKKMeans$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.

Examples

```r
learner = mlr3::lrn("clust.kkmeans")
print(learner)

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

---

**mlr_learners_clust.kmeans**

*K-Means Clustering Learner*

**Description**

A `LearnerClust` for k-means clustering implemented in `stats::kmeans()`. `stats::kmeans()` doesn’t have a default value for the number of clusters. Therefore, the centers parameter here is set to 2 by default. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

**Dictionary**

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

```r
mlr_learners$get("clust.kmeans")
lrn("clust.kmeans")
```
Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: mlr3, mlr3cluster, 'stats', clue

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>centers</td>
<td>untyped</td>
<td>2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>iter.max</td>
<td>integer</td>
<td>10</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td>character</td>
<td>Hartigan-Wong</td>
<td>Hartigan-Wong, Lloyd, Forgy, MacQueen</td>
<td>-</td>
</tr>
<tr>
<td>nstart</td>
<td>integer</td>
<td>1</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>trace</td>
<td>integer</td>
<td>0</td>
<td>[0, ∞)</td>
<td></td>
</tr>
</tbody>
</table>

Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustKMeans

Methods

Public methods:
- LearnerClustKMeans$new()
- LearnerClustKMeans$clone()

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

Usage:
LearnerClustKMeans$new()

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

Usage:
LearnerClustKMeans$clone(deep = FALSE)

Arguments:
deept Whether to make a deep clone.

Examples

learner = mlr3::lrn("clust.kmeans")
print(learner)

# available parameters:
learner$param_set$ids()
Description

A LearnerClust for Mean Shift clustering implemented in LPCM::ms(). There is no predict method for LPCM::ms(), so the method returns cluster labels for the 'training' data.

Dictionary

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

```
mlr_learners$get("clust.meanshift")
lrn("clust.meanshift")
```

Meta Information

- Task type: "clust"
- Predict Types: "partition"
- Feature Types: "logical", "integer", "numeric"
- Required Packages: mlr3, mlr3cluster, LPCM

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>untyped</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>subset</td>
<td>untyped</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>scaled</td>
<td>integer</td>
<td>1</td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>iter</td>
<td>integer</td>
<td>200</td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>thr</td>
<td>numeric</td>
<td>0.01</td>
<td>(-∞, ∞)</td>
</tr>
</tbody>
</table>

Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustMeanShift

Methods

- LearnerClustMeanShift$new()
- LearnerClustMeanShift$clone()
**Method** `new()`: Creates a new instance of this R6 class.

*Usage:*

```r
class(LearnerClustMeanShift)$new()
```

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

*Usage:*

```r
class(LearnerClustMeanShift)$clone(deep = FALSE)
```

*Arguments:*

- `deep` Whether to make a deep clone.

**Examples**

```r
learner = mlr3::lrn("clust.meanshift")
print(learner)

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

---

**mlr_learners_clust.MiniBatchKMeans**

*Mini Batch K-Means Clustering Learner*

**Description**

A LearnerClust for mini batch k-means clustering implemented in `ClusterR::MiniBatchKmeans()`. `ClusterR::MiniBatchKmeans()` doesn’t have a default value for the number of clusters. Therefore, the `clusters` parameter here is set to 2 by default. The predict method uses `ClusterR::predict_MBatchKMeans()` to compute the cluster memberships for new data. The learner supports both partitional and fuzzy clustering.

**Dictionary**

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

```r
mlr_learners$get("clust.MBatchKMeans")
lrn("clust.MBatchKMeans")
```

**Meta Information**

- Task type: “clust”
- Predict Types: “partition”, “prob”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3, mlr3cluster, ClusterR`

**Parameters**
### Super classes

`mlr3::Learner` <- `mlr3cluster::LearnerClust` <- `LearnerClustMiniBatchKMeans`

### Methods

**Public methods:**

- `LearnerClustMiniBatchKMeans$new()`  
- `LearnerClustMiniBatchKMeans$clone()`

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

**Usage:**

```r
LearnerClustMiniBatchKMeans$new()
```

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

**Usage:**

```r
LearnerClustMiniBatchKMeans$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.

### Examples

```r
learner = mlr3::lrn("clust.MBatchKMeans")
print(learner)

# available parameters:
learner$param_set$ids()
```
Partitioning Around Medoids Clustering Learner

Description

A LearnerClust for PAM clustering implemented in `cluster::pam()`. `cluster::pam()` doesn’t have a default value for the number of clusters. Therefore, the \( k \) parameter which corresponds to the number of clusters here is set to 2 by default. The predict method uses `clue::cl_predict()` to compute the cluster memberships for new data.

Dictionary

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

```r
mlr_learners$get("clust.pam")
lrn("clust.pam")
```

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: `mlr3`, `mlr3cluster`, `cluster`

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>integer</td>
<td>2</td>
<td>[1, ( \infty ))</td>
<td></td>
</tr>
<tr>
<td>metric</td>
<td>character</td>
<td>-</td>
<td>euclidian, manhattan</td>
<td>-</td>
</tr>
<tr>
<td>medoids</td>
<td>untyped</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>stand</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>do.swap</td>
<td>logical</td>
<td>TRUE</td>
<td>TRUE, FALSE</td>
<td>-</td>
</tr>
<tr>
<td>pamonce</td>
<td>integer</td>
<td>0</td>
<td>[0, 5]</td>
<td></td>
</tr>
<tr>
<td>trace.lev</td>
<td>integer</td>
<td>0</td>
<td>[0, ( \infty ))</td>
<td></td>
</tr>
</tbody>
</table>

Super classes

`mlr3::Learner` -> `mlr3cluster::LearnerClust` -> `LearnerClustPAM`
Methods

Public methods:

• LearnerClustPAM$new()
• LearnerClustPAM$clone()

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

Usage:
LearnerClustPAM$new()

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

Usage:
LearnerClustPAM$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Examples

learner = mlr3::lrn("clust.pam")
print(learner)

# available parameters:
learner$param_set$ids()

Description

A LearnerClust for Simple K Means clustering implemented in RWeka::SimpleKMeans(). The predict method uses RWeka::predict.Weka_clusterer() to compute the cluster memberships for new data.

Dictionary

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

mlr_learners$get("clust.SimpleKMeans")
lrn("clust.SimpleKMeans")

Meta Information

• Task type: “clust”
• Predict Types: “partition”
• Feature Types: “logical”, “integer”, “numeric”
• Required Packages: mlr3, mlr3cluster, RWeka
Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>untyped</td>
<td>weka.core.EuclideanDistance</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>fast</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>integer</td>
<td>100</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>init</td>
<td>integer</td>
<td>0</td>
<td>[0, 3]</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>max_candidates</td>
<td>integer</td>
<td>100</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>min_density</td>
<td>integer</td>
<td>2</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>integer</td>
<td>2</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>num_slots</td>
<td>integer</td>
<td>1</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>periodic_pruning</td>
<td>integer</td>
<td>10000</td>
<td>[1, ∞)</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>integer</td>
<td>10</td>
<td>[0, ∞)</td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>numeric</td>
<td>-1</td>
<td>(-∞, ∞)</td>
<td></td>
</tr>
<tr>
<td>t1</td>
<td>numeric</td>
<td>-1.5</td>
<td>(-∞, ∞)</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
<tr>
<td>output_debug_info</td>
<td>logical</td>
<td>FALSE</td>
<td>TRUE, FALSE</td>
<td></td>
</tr>
</tbody>
</table>

Super classes

mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustSimpleKMeans

Methods

Public methods:
- LearnerClustSimpleKMeans$new()
- LearnerClustSimpleKMeans$clone()

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

Usage:
LearnerClustSimpleKMeans$new()

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

Usage:
LearnerClustSimpleKMeans$clone(deep = FALSE)

Arguments:
deepe Whether to make a deep clone.
Examples

learner = mlr3::lrn("clust.SimpleKMeans")
print(learner)

# available parameters:
learner$param_set$ids()

mlr_learners_clust.xmeans

X-means Clustering Learner

Description

A LearnerClust for X-means clustering implemented in RWeka::XMeans(). The predict method uses RWeka::predict.Weka_clusterer() to compute the cluster memberships for new data.

Dictionary

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

mlr_learners$get("clust.xmeans")
lrn("clust.xmeans")

Meta Information

- Task type: “clust”
- Predict Types: “partition”
- Feature Types: “logical”, “integer”, “numeric”
- Required Packages: mlr3, mlr3cluster, RWeka

Parameters

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
<th>Default</th>
<th>Levels</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>numeric</td>
<td>1</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>C</td>
<td>numeric</td>
<td>0</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>D</td>
<td>untyped</td>
<td>weka.core.EuclideanDistance</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>H</td>
<td>integer</td>
<td>4</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>I</td>
<td>integer</td>
<td>1</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>J</td>
<td>integer</td>
<td>1000</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>K</td>
<td>untyped</td>
<td></td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>L</td>
<td>integer</td>
<td>2</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>M</td>
<td>integer</td>
<td>1000</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>S</td>
<td>integer</td>
<td>10</td>
<td></td>
<td>[1, ∞)</td>
</tr>
<tr>
<td>U</td>
<td>integer</td>
<td>0</td>
<td></td>
<td>[0, ∞)</td>
</tr>
<tr>
<td>use_kdtree</td>
<td>logical</td>
<td>FALSE</td>
<td></td>
<td>TRUE, FALSE</td>
</tr>
</tbody>
</table>


Super classes

```
mlr3::Learner -> mlr3cluster::LearnerClust -> LearnerClustXMeans
```

Methods

**Public methods:**

- `LearnerClustXMeans$new()`
- `LearnerClustXMeans$clone()`

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

**Usage:**

```
LearnerClustXMeans$new()
```

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

**Usage:**

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

**Arguments:**

- `deep` Whether to make a deep clone.

**Examples**

```
learner = mlr3::lrn("clust.xmeans")
print(learner)

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

---

**Description**

The score function calls `clusterCrit::intCriteria()` from package `clusterCrit`. Argument `crit` is set to "Calinski_Harabasz".

The score function calls `clusterCrit::intCriteria()` from package `clusterCrit`. Argument `crit` is set to "Calinski_Harabasz".
Format

\texttt{R6::R6Class()} inheriting from \texttt{MeasureClust}.
\texttt{R6::R6Class()} inheriting from \texttt{MeasureClust}.

Construction

This measures can be retrieved from the dictionary \texttt{mlr_measures}:

```
mlr_measures$get("clust.ch")
msr("clust.ch")
```

This measures can be retrieved from the dictionary \texttt{mlr_measures}:

```
mlr_measures$get("clust.ch")
msr("clust.ch")
```

Meta Information

- Range: $[0, \infty)$
- Minimize: FALSE
- Required predict type: partition

- Range: $[0, \infty)$
- Minimize: FALSE
- Required predict type: partition

See Also

Dictionary of Measures: \texttt{mlr3::mlr_measures}
as.data.table(mlr_measures) for a complete table of all (also dynamically created) \texttt{mlr3::Measure} implementations.

Dictionary of Measures: \texttt{mlr3::mlr_measures}
as.data.table(mlr_measures) for a complete table of all (also dynamically created) \texttt{mlr3::Measure} implementations.

Other cluster measures: \texttt{mlr_measures_clust.db, mlr_measures_clust.dunn, mlr_measures_clust.silhouette, mlr_measures_clust.wss}

Other cluster measures: \texttt{mlr_measures_clust.db, mlr_measures_clust.dunn, mlr_measures_clust.silhouette, mlr_measures_clust.wss}
Description

The score function calls `clusterCrit::intCriteria()` from package `clusterCrit`. Argument `crit` is set to "Davies_Bouldin".

Format

`R6::R6Class()` inheriting from `MeasureClust`.

Construction

This measures can be retrieved from the dictionary `mlr_measures`:

```r
mlr_measures$get("clust.db")
msr("clust.db")
```

Meta Information

- Range: \([0, \infty)\)
- Minimize: `TRUE`
- Required predict type: `partition`

See Also

Dictionary of Measures: `mlr3::mlr_measures`
as.data.table(mlr_measures) for a complete table of all (also dynamically created) `mlr3::Measure` implementations.

Other cluster measures: `mlr_measures_clust.ch, mlr_measures_clust.dunn, mlr_measures_clust.silhouette, mlr_measures_clust.wss`

Description

The score function calls `clusterCrit::intCriteria()` from package `clusterCrit`. Argument `crit` is set to "Dunn".

Format

`R6::R6Class()` inheriting from `MeasureClust`. 
Rousseeuw's Silhouette Quality Index

Description

The score function calls `clusterCrit::intCriteria()` from package `clusterCrit`. Argument `crit` is set to "Silhouette".

Format

`R6::R6Class()` inheriting from `MeasureClust`.

Construction

This measures can be retrieved from the dictionary `mlr_measures`:

```r
mlr_measures$get("clust.silhouette")
msr("clust.silhouette")
```

Meta Information

- Range: `[0, ∞)`
- Minimize: `FALSE`
- Required predict type: `partition`
**See Also**

Dictionary of Measures: mlr3::mlr_measures

as.data.table(mlr_measures) for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: mlr_measures_clust.ch, mlr_measures_clust.db, mlr_measures_clust.dunn, mlr_measures_clust.wss

---

**mlr_measures_clust.wss**

*Within Sum of Squares*

**Description**

The score function calls clusterCrit::intCriteria() from package clusterCrit. Argument crit is set to "Trace_W".

**Format**

R6::R6Class() inheriting from MeasureClust.

**Construction**

This measure can be retrieved from the dictionary mlr_measures:

```r
mlr_measures$get("clust.wss")
msr("clust.wss")
```

**Meta Information**

- Range: $[0, \infty)$
- Minimize: TRUE
- Required predict type: partition

**See Also**

Dictionary of Measures: mlr3::mlr_measures

as.data.table(mlr_measures) for a complete table of all (also dynamically created) mlr3::Measure implementations.

Other cluster measures: mlr_measures_clust.ch, mlr_measures_clust.db, mlr_measures_clust.dunn, mlr_measures_clust.silhouette
Description

A cluster task for the datasets::USArrests data set.

Format

R6::R6Class inheriting from TaskClust.

Construction

mlr_tasks$get("usarrests")

PredictionClust Prediction Object for Cluster Analysis

Description

This object wraps the predictions returned by a learner of class LearnerClust, i.e. the predicted partition and cluster probability.

Super class

mlr3::Prediction -> PredictionClust

Active bindings

partition (integer())
Access the stored partition.

prob (matrix())
Access to the stored probabilities.

Methods

Public methods:

• PredictionClust$new()
• PredictionClust$clone()

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

Usage:
PredictionClust$new(
  task = NULL,
  row_ids = task$row_ids,
  partition = NULL,
  prob = NULL,
  check = TRUE
)

Arguments:

  task (TaskClust)
    Task, used to extract defaults for row_ids.

  row_ids (integer())
    Row ids of the predicted observations, i.e. the row ids of the test set.

  partition (integer())
    Vector of cluster partitions.

  prob (matrix())
    Numeric matrix of cluster membership probabilities with one column for each cluster and
    one row for each observation. Columns must be named with cluster numbers, row names are
    automatically removed. If prob is provided, but partition is not, the cluster memberships
    are calculated from the probabilities using max.col() with ties.method set to "first".

  check (logical(1))
    If TRUE, performs some argument checks and predict type conversions.

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

Usage:
PredictionClust$clone(deep = FALSE)

Arguments:

  deep Whether to make a deep clone.

Examples

library(mlr3)
library(mlr3cluster)
task = tsk("usarrests")
learner = lrn("clust.kmeans")
p = learner$train(task)$predict(task)
p$predict_types
head(as.data.table(p))

TaskClust            Cluster Task

Description

This task specializes mlr3::Task for cluster problems. As an unsupervised task, this task has no
target column. The task_type is set to "clust".

Predefined tasks are stored in the dictionary mlr_tasks.
Super classes

\texttt{mlr3::Task} $\rightarrow$ \texttt{mlr3::TaskUnsupervised} $\rightarrow$ TaskClust

Methods

Public methods:

\begin{itemize}
  \item TaskClust$new()
  \item TaskClust$clone()
\end{itemize}

Method \texttt{new()}: Creates a new instance of this \texttt{R6} class.

\textit{Usage}:

\texttt{TaskClust$new(id, backend, label = NA\_character\_)}

\textit{Arguments}:

\begin{itemize}
  \item \texttt{id (character(1))} Identifier for the new instance.
  \item \texttt{backend (DataBackend)} Either a DataBackend, or any object which is convertible to a DataBackend with as\_data\_backend(). E.g., a data.frame() will be converted to a DataBackend\_DataTable.
  \item \texttt{label (character(1))} Label for the new instance.
\end{itemize}

Method \texttt{clone()}: The objects of this class are cloneable with this method.

\textit{Usage}:

\texttt{TaskClust$clone(deep = FALSE)}

\textit{Arguments}:

\begin{itemize}
  \item \texttt{deep} Whether to make a deep clone.
\end{itemize}

Examples

\begin{verbatim}
library(mlr3)
library(mlr3cluster)
task = TaskClust$new("usarrests", backend = USArrests)
task$task_type

# possible properties:
mlr_reflections$task_properties$clust
\end{verbatim}
Index

* Prediction
  PredictionClust, 42

* Task
  TaskClust, 43

* cluster measures
  mlr_measures_clust.ch, 37
  mlr_measures_clust.db, 39
  mlr_measures_clust.dunn, 39
  mlr_measures_clust.silhouette, 40
  mlr_measures_clust.wss, 41

apcluster::apcluster(), 10
as_prediction_clust, 3
as_task_clust, 4

clue::cl_predict(), 13, 28, 33
clust.dunn, 8
cluster::agnes(), 9
cluster::diana(), 18
cluster::fanny(), 21
cluster::pam(), 33
clusterCrit::intCriteria(), 37, 39–41
ClusterR::MiniBatchKmeans(), 31
ClusterR::predict_MBatchKMeans(), 31
data.frame(), 4
DataBackend, 4, 44
DataBackendDataTable, 44
datasets::USArrests, 42
dbscan::dbscan(), 16
dbscan::predict.dbscan_fast(), 16
Dictionary, 38–41
dictionary, 9, 10, 13, 14, 17–19, 21, 22, 24, 25, 27, 28, 30, 31, 33, 34, 36, 43
e1071::cmeans(), 13
kernlab::kkmeans(), 27
Learner, 6, 8–10, 13, 14, 17–19, 21, 22, 24, 25, 27, 28, 30, 31, 33, 34, 36
LearnerClust, 5, 9, 10, 13, 14, 16, 18, 19, 21, 22, 24, 25, 27, 28, 30, 31, 33, 34, 36, 42
LearnerClustAgnes
  (mlr_learners_clust.agnes), 9
LearnerClustAP (mlr_learners_clust.ap), 10
LearnerClustCMeans
  (mlr_learners_clust.cmeans), 13
LearnerClustCobweb
  (mlr_learners_clust.cobweb), 14
LearnerClustDBSCAN
  (mlr_learners_clust.dbscan), 16
LearnerClustDiana
  (mlr_learners_clust.diana), 18
LearnerClustEM (mlr_learners_clust.em), 19
LearnerClustFanny
  (mlr_learners_clust.fanny), 21
LearnerClustFarthestFirst
  (mlr_learners_clust.FF), 24
LearnerClustFeatureless
  (mlr_learners_clust.featureless), 22
LearnerClustHclust
  (mlr_learners_clust.hclust), 25
LearnerClustKKMeans
  (mlr_learners_clust.kkmeans), 27
LearnerClustKMeans
  (mlr_learners_clust.kmeans), 28
LearnerClustMeanShift
  (mlr_learners_clust.meanshift), 30
LearnerClustMiniBatchKMeans
  (mlr_learners_clust.MiniBatchKMeans), 31
LearnerClustPAM
  (mlr_learners_clust.pam), 33
LearnerClustSimpleKMeans
  (mlr_learners_clust.SimpleKMeans), 34
LearnerClustKMeans
  (mlr_learners_clust.KMeans), 36
LPCM::ms(), 30
lrn(), 9, 10, 13, 14, 17–19, 21, 22, 24, 25, 27, 28, 30, 31, 33, 34, 36
max.col(), 43
mean(), 8
MeasureClust, 7, 38–41
Measures, 38–41
mlr3::Learner, 5, 9, 12, 13, 16, 17, 19, 20, 22–24, 26, 28–30, 32, 33, 35, 37
mlr3::Measure, 7, 38–41
mlr3::mlr_learners, 5
mlr3::mlr_measures, 7, 38–41
mlr3::mlr_learners_clust.agnes, 9
mlr3::mlr_learners_clust.ap, 10
mlr3::mlr_learners_clust.cmeans, 13
mlr3::mlr_learners_clust.cobweb, 14
mlr3::mlr_learners_clust.dbscan, 16
mlr3::mlr_learners_clust.diana, 18
mlr3::mlr_learners_clust.em, 19
mlr3::mlr_learners_clust.fanny, 21
mlr3::mlr_learners_clust.featureless, 22
mlr3::mlr_learners_clust.FF, 24
mlr3::mlr_learners_clust.hclust, 25
mlr3::mlr_learners_clust.kkmeans, 27
mlr3::mlr_learners_clust.kmeans, 28
mlr3::mlr_learners_clust.meanshift, 30
mlr3::mlr_learners_clust.MiniBatchKMeans, 31
mlr3::mlr_learners_clust.pam, 33
mlr3::mlr_learners_clust.SimpleKMeans, 34
mlr3::mlr_learners_clust.xmeans, 36
mlr_measures, 38–41
mlr_measures_clust.db, 38, 39, 40, 41
mlr_measures_clust.dunn, 38, 39, 40, 41
mlr_measures_clust.silhouette, 38–40, 41
mlr_measures_clust.wss, 38–41, 41
mlr_reflections$learner_predict_types, 6, 8
mlr_reflections$learner_properties, 6
mlr_reflections$measure_properties, 8
mlr_reflections$task_feature_types, 6
mlr_tasks, 43
mlr_tasks_usarrests, 42
paradox::ParamSet, 6
Prediction, 5
PredictionClust, 3, 5, 42
R6, 6, 7, 10, 12, 14, 16, 17, 19, 20, 22–24, 26, 28, 29, 31, 32, 34, 35, 37, 42, 44
R6::R6Class, 42
R6::R6Class(), 38–41
requireNamespace(), 6, 8
Resampling, 8
RWeka::Cobweb(), 14
RWeka::FarthestFirst(), 24
RWeka::list_Weka_interfaces(), 19
RWeka::predict.Weka_clusterer(), 14, 19, 24, 34, 36
RWeka::SimpleKMeans(), 34
RWeka::XMeans(), 36
stats::cutree(), 9, 18
stats::dist(), 25
stats::hclust(), 25
stats::kmeans(), 28
Task, 8
TaskClust, 4, 5, 42, 43, 43