Package 'mlr3oml'

September 24, 2021

Title Connector Between 'mlr3' and 'OpenML'

Version 0.5.0

Description Provides an interface to 'OpenML.org' to list and download machine learning data and tasks. Data and tasks can be automatically converted to 'mlr3' tasks. For a more sophisticated interface which also allows uploading experiments, see the 'OpenML' package.

License LGPL-3

URL https://mlr3oml.mlr-org.com, https://github.com/mlr-org/mlr3oml

BugReports https://github.com/mlr-org/mlr3oml

Depends R (>= 3.1.0)

Imports backports (>= 1.1.6), checkmate, curl, data.table, jsonlite, lgr, mlr3 (>= 0.10.0), mlr3misc (>= 0.7.0), R6, stringi

Suggests RWeka, farff, foreign, mlr3proba, qs, testthat (>= 3.0.0), withr

Encoding UTF-8

NeedsCompilation yes

Config/testthat/edition 3

RoxygenNote 7.1.2

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

Maintainer Michel Lang <michellang@gmail.com>

Repository CRAN

Date/Publication 2021-09-24 09:50:02 UTC

R topics documented:

`mlr3oml-package` *mlr3oml: Connector Between 'mlr3' and 'OpenML'*

Description

Provides an interface to 'OpenML.org' to list and download machine learning data and tasks. Data and tasks can be automatically converted to 'mlr3' tasks. For a more sophisticated interface which also allows uploading experiments, see the 'OpenML' package.

mlr3 Integration

This package adds the mlr3::Task `"oml"` and the mlr3::Resampling `"oml"` to mlr3::mlr_tasks and mlr3::mlr_resamplings, respectively. For the former you may pass either a `data_id` or a `task_id`, the latter requires a `task_id`.

Options

- `mlr3oml.cache`: Enables or disables caching globally. If set to `FALSE`, caching is disabled. If set to `TRUE`, cache directory as reported by `R_user_dir()` is used. Alternatively, you can specify a path on the local file system here. Default is `FALSE`.
- `mlr3oml.api_key`: API key to use. All operations supported by this package work without an API key, but you might get rate limited without an API key. If not set, defaults to the value of the environment variable `OPENMLAPIKEY`.
- `mlr3oml.arff_parser`: ARFF parser to use, defaults to the internal one relies on `data.table::fread()`. Can also be set to `"RWeka"` for the parser in **RWeka** or `"farff"` for the reader implemented in **farff**.

Logging

The **lgr** package is used for logging. To change the threshold, use `lgr::get_logger("mlr3oml")$set_threshold()`.

Author(s)

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

See Also

Useful links:

- https://mlr3oml.mlr-org.com
- https://github.com/mlr-org/mlr3oml
- Report bugs at https://github.com/mlr-org/mlr3oml

list_oml *List Data from OpenML*

Description

This function allows to query data sets, tasks, flows, setups, runs, and evaluation measures from https://openml.org/d using some simple filter criteria.

Usage

```
list_oml_data_sets(
  data_id = NULL,
  data_name = NULL,
  number_instances = NULL,
  number_features = NULL,
  number_classes = NULL,
  number_missing_values = NULL,
  tag = NULL,
  limit = getOption("mlr3oml.limit", 5000L),
  ...
)

list_oml_evaluations(
  run_id = NULL,
  task_id = NULL,
  measures = NULL,
  tag = NULL,
  limit = getOption("mlr3oml.limit", 5000L),
  ...
)

list_oml_flows(
  uploader = NULL,
  tag = NULL,
  limit = getOption("mlr3oml.limit", 5000L),
  ...
)

list_oml_measures()

list_oml_runs(
  run_id = NULL,
  task_id = NULL,
  tag = NULL,
  limit = getOption("mlr3oml.limit", 5000L),
  ...
)
```

```
list_oml_setups(
  flow_id = NULL,
  setup_id = NULL,
  tag = NULL,
  limit = getOption("mlr3oml.limit", 5000L),
  ...
)

list_oml_tasks(
  task_id = NULL,
  data_id = NULL,
  number_instances = NULL,
  number_features = NULL,
  number_classes = NULL,
  number_missing_values = NULL,
  tag = NULL,
  limit = getOption("mlr3oml.limit", 5000L),
  ...
)
```

Arguments

`data_id`	(`integer()`) Vector of data ids to restrict to.
`data_name`	(`character(1)`) Filter for name of data set.
`number_instances`	(`integer()`) Filter for number of instances.
`number_features`	(`integer()`) Filter for number of features.
`number_classes`	(`integer()`) Filter for number of labels of the target (only classification tasks).
`number_missing_values`	(`integer()`) Filter for number of missing values.
`tag`	(`character()`) Filter for tags. You can provide multiple tags as character vector.
`limit`	(`integer()`) Limit the results to `limit` records. Default is the value of option `"mlr3oml.limit"`, defaulting to 5000.
`...`	(any) Additional (unsupported) filters, as named arguments.
`run_id`	(`integer()`) Vector of run ids to restrict to.

`task_id` (`integer()`)
Vector of task ids to restrict to.

`measures` (`character()`)
Vector of evaluation measures to restrict to.

`uploader` (`integer(1)`)
Filter for uploader.

`flow_id` (`integer(1)`)
Filter for flow id.

`setup_id` (`integer()`)
Vector of setup ids to restrict to.

Details

Filter values are usually provided as single atomic values (typically integer or character). Provide a numeric vector of length 2 (`c(l,u)`) to find matches in the range $[l, u]$.

Note that only a subset of filters is exposed here. For a more feature-complete package, see **OpenML**. Alternatively, you can pass additional filters via `...` using the names of the official API, c.f. https://www.openml.org/api_docs.

Value

(`data.table()`) of results, or a null data.table if no data set matches the filter criteria.

References

Casalicchio G, Bossek J, Lang M, Kirchhoff D, Kerschke P, Hofner B, Seibold H, Vanschoren J, Bischl B (2017). "OpenML: An R Package to Connect to the Machine Learning Platform OpenML." *Computational Statistics*, 1–15. doi: 10.1007/s0018001707422.

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi: 10.1145/2641190.2641198.

Examples

```
### query data sets
# search for titanic data set
data_sets = list_oml_data_sets(data_name = "titanic")
print(data_sets)

# search for a reduced version
data_sets = list_oml_data_sets(
  data_name = "titanic",
  number_instances = c(2200, 2300),
  number_features = 4
)
print(data_sets)

### search tasks for this data set
tasks = list_oml_tasks(data_id = data_sets$data_id)
```

```
print(tasks)


# query runs, group by number of runs per task_id
runs = list_oml_runs(task_id = tasks$task_id)
runs[, .N, by = task_id]
```

`OMLData` *Interface to OpenML Data Sets*

Description

This is the class for data sets served on https://openml.org/d.

mlr3 Integration

A mlr3::Task is returned by the method $task. Alternatively, you can convert this object to a mlr3::DataBackend using `mlr3::as_data_backend()`.

ARFF Files

This package comes with an own reader for ARFF files, based on `data.table::fread()`. For sparse ARFF files and if the **RWeka** package is installed, the reader automatically falls back to the implementation in (`RWeka::read.arff()`).

Public fields

`id (integer(1))`
: OpenML data id.

`cache_dir (logical(1) | character(1))`
: Stores the location of the cache for objects retrieved from https://openml.org. If set to `FALSE`, caching is disabled.

  The package **qs** is required for caching.

Active bindings

`name (character(1))`
: Name of the data set, as extracted from the data set description.

`desc (list())`
: Data set description (meta information), downloaded and converted from the JSON API response.

`qualities (data.table())`
: Data set qualities (performance values), downloaded from the JSON API response and converted to a `data.table::data.table()` with columns `"name"` and `"value"`.

`features (data.table())`
: Information about data set features (including target), downloaded from the JSON API response and converted to a `data.table::data.table()` with columns:
  - `"index"` (`integer()`): Column position.
  - `"name"` (`character()`): Name of the feature.
  - `"data_type"` (`factor()`): Type of the feature: `"nominal"` or `"numeric"`.
  - `"nominal_value"` (`list()`): Levels of the feature, or `NULL` for numeric features.
  - `"is_target"` (`logical()`): `TRUE` for target column, `FALSE` otherwise.
  - `"is_ignore"` (`logical()`): `TRUE` if this feature should be ignored. Ignored features are removed automatically from the data set.
  - `"is_row_identifier"` (`logical()`): `TRUE` if the column encodes a row identifier. Row identifiers are removed automatically from the data set.
  - `"number_of_missing_values"` (`integer()`): Number of missing values in the column.

`data (data.table())`
: Data as `data.table::data.table()`. Columns marked as row identifiers or marked with the ignore flag are automatically removed.

`target_names (character())`
: Name of the default target, as extracted from the OpenML data set description.

`feature_names (character())`
: Name of the features, as extracted from the OpenML data set description.

`nrow (integer())`
: Number of observations, as extracted from the OpenML data set qualities.

`ncol (integer())`
: Number of features (including targets), as extracted from the table of data set features. This excludes row identifiers and ignored columns.

`tags (character())`
: Returns all tags of the data set.

Methods

Public methods:

- `OMLData$new()`
- `OMLData$print()`
- `OMLData$quality()`
- `OMLData$task()`
- `OMLData$clone()`

Method `new()`**:** Creates a new object of class `OMLData`.

Usage:

```
OMLData$new(id, cache = getOption("mlr3oml.cache", FALSE))
```

Arguments:

`id (integer(1))`
: OpenML data id.

`cache (logical(1) | character(1))`
See field cache for an explanation of possible values. Defaults to value of option `"mlr3oml.cache"`, or `FALSE` if not set.

Method `print()`**:** Prints the object. For a more detailed printer, convert to a mlr3::Task via $task().

Usage:

```
OMLData$print()
```

Method `quality()`**:** Returns the value of a single OpenML data set quality.

Usage:

```
OMLData$quality(name)
```

Arguments:

`name (character(1))`
Name of the quality to extract.

Method `task()`**:** Creates a mlr3::Task using the provided target column, defaulting to the default target attribute of the task description. Note that if the target column is incorrectly encoded, e.g. as numeric 0/1 for classification, this will result in a task of the wrong type.

Usage:

```
OMLData$task(target_names = NULL)
```

Arguments:

`target_names (character())`
Name(s) of the target columns, or `NULL` for the default columns.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi: 10.1145/2641190.2641198.

Examples

```
odata = OMLData$new(id = 9)

print(odata)
print(odata$target_names)
print(odata$feature_names)
print(odata$tags)
print(odata$task())
```

```
# get a task via tsk():
if (requireNamespace("mlr3")) {
  mlr3::tsk("oml", data_id = 9)
}
```

`OMLTask` *Interface to OpenML Tasks*

Description

This is the class for tasks served on https://openml.org/t.

mlr3 Integration

A mlr3::Task is returned by the method $task. Alternatively, you can convert this object to a mlr3::DataBackend using `mlr3::as_data_backend()`.

Public fields

`id (integer(1))`
: OpenML task id.

`cache_dir (logical(1) | character(1))`
: Stores the location of the cache for objects retrieved from https://openml.org. If set to `FALSE`, caching is disabled.

The package **qs** is required for caching.

Active bindings

`name (character(1))`
: Name of the task, as extracted from the task description.

`desc (list())`
: Task description (meta information), downloaded and converted from the JSON API response.

`data_id (integer())`
: Data id, extracted from the task description.

`data (OMLData)`
: Access to the underlying OpenML data set via a OMLData object.

`nrow (integer())`
: Number of rows, as extracted from the OMLData object.

`ncol (integer())`
: Number of columns, as extracted from the OMLData object.

`target_names (character())`
: Name of the targets, as extracted from the OpenML task description.

`feature_names (character())`
: Name of the features (without targets of this OMLTask).

`task` (mlr3::Task)
Creates a mlr3::Task using the target attribute of the task description.

`resampling` (mlr3::Resampling)
Creates a ResamplingCustom using the target attribute of the task description.

`tags` (`character()`)
Returns all tags of the task.

Methods

Public methods:

- `OMLTask$new()`
- `OMLTask$print()`
- `OMLTask$clone()`

Method `new()`**:** Creates a new object of class `OMLTask`.

Usage:

```
OMLTask$new(id, cache = getOption("mlr3oml.cache", FALSE))
```

Arguments:

`id` (`integer(1)`)
OpenML task id.

`cache` (`logical(1)|character(1)`)
See field `cache` for an explanation of possible values. Defaults to value of option `"mlr3oml.cache"`, or `FALSE` if not set.

Method `print()`**:** Prints the object. For a more detailed printer, convert to a mlr3::Task via $task.

Usage:

```
OMLTask$print()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Vanschoren J, van Rijn JN, Bischl B, Torgo L (2014). "OpenML." *ACM SIGKDD Explorations Newsletter*, **15**(2), 49–60. doi: 10.1145/2641190.2641198.

Examples

```
otask = OMLTask$new(id = 59)

print(otask)
print(otask$target_names)
print(otask$feature_names)
print(otask$tags)
print(otask$task)

# get a task via tsk():
if (requireNamespace("mlr3")) {
  mlr3::tsk("oml", task_id = 59)
}
```

---

read_arff *Read ARFF file*

Description

Parses a file located at path and returns a data.table().

Usage

```
read_arff(path)
```

Arguments

path	(character(1)) Path or URI of the ARFF file, passed to file().

Value

(data.table()).

Index