

Package: mlr3hf (via r-universe)

September 27, 2026

Title Convert Hugging Face Datasets into mlr3 Tasks

Version 0.0.0.9000

Description Provides a simple interface to download tabular datasets from the Hugging Face Hub and convert them into mlr3 Task objects.

License MIT + file LICENSE

URL <https://mlr-org.github.io/mlr3hf/>,
<https://github.com/mlr-org/mlr3hf>

BugReports <https://github.com/mlr-org/mlr3hf/issues>

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/Needs/website pkgdown

Imports httr, jsonlite, data.table, nanoparquet, fs, R6, mlr3, mlr3misc, filelock, rappdirs, checkmate, curl, glue, cli

Suggests webmockr, vcr, rmarkdown, knitr, testthat (>= 3.0.0), arrow, withr, xgboost, mlr3pipelines, mlr3learners, rpart

VignetteBuilder knitr

Config/pak/sysreqs cmake make libuv1-dev libssl-dev

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

Date/Publication 2026-08-28 10:48:15 UTC

RemoteUrl <https://github.com/mlr-org/mlr3hf>

RemoteRef HEAD

RemoteSha f9d97751747615fc56c03d5d987a0dafcfad7f2a

Contents

as_data_backend.HFData	2
as_task.HFData	3
cache_hfhub	3
cache_parquet	5
download_desc	6
fetch_dataset_meta	7
HFData	7
hfdt	10
list_datasets	11
mlr3hf	12
Index	14

as_data_backend.HFData	<i>Coerce an HFData Object to an mlr3 Data Backend</i>
------------------------	--

Description

Converts an [HFData](#) object into an `mlr3` [DataBackend](#), triggering download and parsing of the underlying dataset if this has not already happened.

Usage

```
## S3 method for class 'HFData'
as_data_backend(data, ...)
```

Arguments

- `data` (HFData)
The HFData object to convert.
- `...` Additional arguments (currently unused).

Value

An `mlr3` [DataBackend](#).

as_task.HFData	<i>Coerce an HFData Object to an mlr3 Task</i>
----------------	--

Description

Converts an [HFData](#) object into an [mlr3 Task](#), either a [TaskClassif](#) or a [TaskRegr](#). The target column and task type can be taken from the HFData object or overridden explicitly.

When `task_type = "auto"`, the task type is inferred from the class of the target column: factor and logical columns become classification tasks, numeric and integer columns become regression tasks.

Usage

```
## S3 method for class 'HFData'
as_task(x, target_names = NULL, task_type = NULL, ...)
```

Arguments

<code>x</code>	(HFData) The HFData object to convert.
<code>target_names</code>	(character()) Name of the target column. If NULL, the target stored in x is used. Multiple targets are not supported.
<code>task_type</code>	(character(1)) One of "auto", "classif", or "regr". Defaults to "auto", in which case the type is inferred from the target column.
<code>...</code>	Additional arguments (currently unused).

Value

An [mlr3::TaskClassif](#) or [mlr3::TaskRegr](#) object.

cache_hfhub	<i>Cache a File from a Hugging Face Hub Repository</i>
-------------	--

Description

Downloads and caches a specific file from the canonical (original) version of a dataset repository on the Hugging Face Hub — that is, the dataset as published by its authors, rather than the auto-generated Parquet conversion (see [cache_parquet](#) for that). If the file has already been cached locally, it is reused instead of being downloaded again.

Usage

```
cache_hfhub(
  repo_id,
  file_name,
  local_files_only = FALSE,
  revision = "main",
  ...
)
```

Arguments

<code>repo_id</code>	Character string. The repository ID of the dataset on the Hugging Face Hub, in the form "user_id/repo_name" (e.g. "scikit-learn/iris").
<code>file_name</code>	Character string. The name of the file to download from the repository, e.g. "iris.csv".
<code>local_files_only</code>	Logical. If TRUE, only the local cache is checked and no network request is made; an error is raised if the file is not already cached. If FALSE (default), the file is downloaded if not already present, or if a newer revision is available.
<code>revision</code>	Character string. The branch, tag, or commit hash to download the file from. Defaults to "main".
<code>...</code>	Additional arguments for future use.

Details

Files are cached in a local directory following the Hugging Face Hub caching convention, keyed by `repo_id`, `revision`, and `file_name`. Subsequent calls with the same arguments will return the cached path without re-downloading, unless the remote file has changed.

Value

Character string giving the local file path where the cached file is stored.

See Also

[cache_parquet](#) for caching the auto-converted Parquet version of a dataset instead.

https://huggingface.co/docs/huggingface_hub/en/guides/manage-cache

Examples

```
cache_hfhub(
  repo_id = "scikit-learn/iris",
  file_name = "iris.csv"
)

# Use a cached copy only, without attempting a network request
cache_hfhub(
  repo_id = "scikit-learn/iris",
```

```

    file_name          = "iris.csv",
    local_files_only = TRUE
)

```

cache_parquet

*Cache a Parquet Export of a Hugging Face Dataset***Description**

Downloads and caches the Parquet files for a dataset hosted on the Hugging Face Hub, using the auto-generated refs/convert/parquet branch. This branch contains a Parquet conversion of the dataset that Hugging Face maintains automatically, which avoids the need to load the dataset via its original (often script-based) loading format.

Usage

```

cache_parquet(
  repo_id,
  config,
  split = NULL,
  revision = "refs%2Fconvert%2Fparquet",
  ...
)

```

Arguments

repo_id	Character string. The repository ID of the dataset on the Hugging Face Hub, in the form "user_id/repo_name" (e.g. "fancyzhx/amazon_polarity").
config	Character string. The dataset configuration name. Many datasets expose only one configuration, in which case "default" can usually be used. If unsure, inspect the repository's refs/convert/parquet branch on the Hub to see the available configuration folders.
split	Character string. The dataset split to cache, such as "train", "test", or "validation". If NULL (default), all available splits for the given configuration are cached.
revision	Character string. A specific commit hash to pin the download to, instead of using the latest revision of the refs/convert/parquet branch. Defaults to NULL, which resolves to the most recent commit.
...	Additional arguments, reserved for future use.

Details

Files are downloaded from the refs/convert/parquet branch maintained by Hugging Face, not from the dataset's default branch. Large datasets may be split across multiple Parquet shards; all matching shards for the requested split are downloaded and cached.

Value

A named list with one entry per cached split. Each entry is a character vector of local file paths to the cached Parquet shard(s) for that split (a single path if the split has one shard, multiple paths if it was sharded across several files).

See Also

<https://huggingface.co/docs/datasets-server/en/parquet>

Examples

```
cache_parquet(
  repo_id = "scikit-learn/iris",
  config = "default"
)

cache_parquet(
  repo_id = "fancyzhx/amazon_polarity",
  config = "amazon_polarity",
  split = "train"
)
```

download_desc

download_desc

Description

it helps in downloading the description

Usage

```
download_desc(repo_id, revision = "main", ..., files_metadata = FALSE)
```

Arguments

repo_id	repo id of the dataset
revision	revision of the dataset, default is "main"
...	additional arguments passed to http::GET
files_metadata	whether to include files metadata, default is FALSE

Value

list containing description and downloads of the dataset

Examples

```
download_desc("scikit-learn/iris")
```

fetch_dataset_meta	<i>Fetch nrows/ncols for a vector of dataset ids concurrently via curl's multi-handle pool.</i>
--------------------	---

Description

A single call to `/size` per dataset is enough. First the `configs[]` array is checked: if the dataset does not have exactly one config, `nrows/ncols` are both NA. If it does have exactly one config, `nrows` is computed by summing `num_rows` across all of that config's entries in the `splits[]` array (rather than trusting the config-level aggregate directly), and `ncols` is read from `num_columns` on one of those splits (column count doesn't vary across splits of the same config).

Usage

```
fetch_dataset_meta(ids, max_concurrent = 10, timeout = 30)
```

Arguments

<code>ids</code>	Character vector of dataset ids, e.g. "google/dataset-one".
<code>max_concurrent</code>	Integer. Max simultaneous connections in the pool.
<code>timeout</code>	Numeric. Per-request timeout in seconds.

Details

If a dataset has zero or more than one config, both `nrows` and `ncols` are set to NA – there's no single unambiguous row/column count to report in that case.

Value

A `data.table` with columns: `id`, `nrows`, `ncols`.

HFDData	<i>Hugging Face Dataset Wrapper</i>
---------	-------------------------------------

Description

An [R6::R6Class](#) that wraps a dataset hosted on the Hugging Face Hub and exposes it as an `mlr3` compatible data backend / task. Data is loaded lazily: no download or parsing happens until fields that require access to the underlying data (e.g. `data`, `nrow`, `ncol`, `colnames`) are accessed for the first time. Results are then cached for the lifetime of the object.

A dataset can be identified either by a config (one of the dataset's predefined configurations, loaded via its parquet export) or by a single `file_name` within the repository. Exactly one of the two must be supplied before data can be retrieved.

All active bindings are read-only; attempting to assign to them raises an error. See the Fields section below for details on each one.

Note on `clone()`: like every R6 object, HFData objects have an inherited `clone(deep = FALSE)` method for copying the object (deep controls whether nested R6 fields are also cloned). It is not re-documented here since it isn't defined in this class's source, but behaves exactly as described in [R6::R6Class](#).

Active bindings

`repo_id` (`character(1)`)
The Hugging Face repository id, e.g. "user/dataset". Read-only.

`config` (`character(1) | NULL`)
The dataset configuration name, if specified. Read-only.

`file_name` (`character(1) | NULL`)
The specific file name within the repository, if specified. Read-only.

`split` (`character() | NULL`)
The dataset split (e.g. "train", "test"), if specified. Read-only.

`target` (`character() | NULL`)
The name of the target column. Read-only.

`desc` (`list()`)
The repository description/metadata, downloaded from the Hub on first access. Read-only.

`storage` (`character(1)`)
Human-readable size of the storage used by the whole repository (not a single file/config). Read-only.

`data` (`data.frame()`)
The full dataset as a data frame. Triggers backend creation on first access. Read-only.

`nrow` (`integer(1)`)
Number of rows in the dataset. Read-only.

`ncol` (`integer(1)`)
Number of columns in the dataset. Read-only.

`colnames` (`character()`)
Column names of the dataset. Read-only.

`siblings` (`character()`)
File names of all sibling files in the repository. Read-only.

`repo_link` (`character(1)`)
URL to the dataset page on the Hugging Face Hub. Read-only.

`configs` (`character()`)
Available dataset configurations, queried from the datasets-server API. Read-only.

`splits` (`character()`)
Available splits for the selected configuration. Read-only.

`coltypes` (`character()`)
Named vector of the (first) class of each column. Read-only.

`task_type` (`character()`)
One of "auto", "classif", or "regr". Read-only.

`feature_names` (`character()`)
Column names excluding the target column. Read-only.

Methods

Public methods:

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

Method `new()`: Create a new HFData object.

Usage:

```
HFData$new(  
  repo_id,  
  config = NULL,  
  file_name = NULL,  
  split = NULL,  
  target = NULL,  
  primary_key = NULL,  
  task_type = c("auto", "classif", "regr"),  
  ...  
)
```

Arguments:

`repo_id` (character(1)) Repository id on the Hugging Face Hub.

`config` (character(1)NULL) Dataset configuration name. Defaults to NULL, meaning no configuration is selected.

`file_name` (character(1)NULL) Specific file name within the repository. Defaults to NULL, meaning no configuration is selected.

`split` (character()NULL) Dataset split to load. Defaults to NULL, meaning no configuration is selected.

`target` (character()) Name of the target column. Defaults to NULL, meaning no configuration is selected.

`primary_key` (character(1)NULL) Name of the primary key column. Defaults to NULL, meaning no configuration is selected.

`task_type` (character(1)NULL) One of "auto", "classif", "regr".

... Additional arguments, stored for later use.

Returns: A new HFData object.

Method `print()`: Print a short summary of the HFData object, including its dimensions, target column, and repository storage size.

Usage:

```
HFData$print()
```

Returns: `self`, invisibly (called for its side effect).

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

hfdt

*Create an HFData object from a Hugging Face dataset***Description**

Creates an [HFData](#) object for a dataset hosted on the Hugging Face Hub. The returned object can be converted to an mlr3 task using `mlr3::as_task()` or accessed through its active bindings.

Usage

```
hfdt(
  repo_id,
  config = NULL,
  file_name = NULL,
  target = NULL,
  primary_key = NULL,
  split = NULL,
  revision = NULL,
  task_type = c("auto", "classif", "regr"),
  ...
)
```

Arguments

<code>repo_id</code>	(character(1)) Repository ID of the dataset on the Hugging Face Hub, e.g. "scikit-learn/iris" or "ibm-research/duorc".
<code>config</code>	(character(1) NULL) Dataset configuration to load. If NULL, the default configuration is used when available.
<code>file_name</code>	(character(1) NULL) Path to a specific dataset file within the repository. This can be used instead of config to load a particular file.
<code>target</code>	(character NULL) Name(s) of the target column(s). If NULL, the target must be specified later before converting to an mlr3 task.
<code>primary_key</code>	(character NULL) Name(s) of the column(s) that uniquely identify each observation.
<code>split</code>	(character(1) NULL) Dataset split to load, such as "train", "test", or "validation". If NULL, all available splits are loaded.
<code>revision</code>	(character(1) NULL) Revision of the dataset to load.
<code>task_type</code>	(character(1)) Type of task for the dataset. One of "auto", "classif", or "regr". If "auto", the task type is inferred from the target column(s).

... Additional arguments passed to `HFDData`.

Value

An `HFDData` object.

list_datasets	<i>List Datasets from Hugging Face Hub</i>
---------------	--

Description

Query datasets hosted on <https://huggingface.co/datasets> via the Hugging Face Hub API, returning metadata such as dataset id, gated status, and download counts.

Usage

```
list_datasets(
  num_of_dataset = 100,
  chunk_size = 100,
  search = NULL,
  author = NULL,
  fetch_meta = TRUE,
  max_concurrent = 10,
  ...
)
```

Arguments

num_of_dataset	(integer(1)) Total number of records to retrieve. Must be a positive integer. Default is 100.
chunk_size	(integer(1)) Number of records to request per API call. Default is 100.
search	(character(1)) Optional search keyword to filter datasets by. Default is NULL.
author	(character(1)) Optional author/organization name to filter datasets by. Default is NULL.
fetch_meta	(logical(1)) If TRUE (default), also fetches nrows and ncols for each dataset via <code>fetch_dataset_meta()</code> . If FALSE, skips this extra network call and returns only id, gated, downloads.
max_concurrent	(integer(1)) Maximum number of simultaneous connections used when fetching metadata. Default is 10.
...	Additional arguments passed on to <code>get_with_retry()</code> , e.g. <code>max_retries</code> .

Value

(data.table()) of results with columns id, nrows, and ncols or an empty data.table() if no datasets are found.

References

<https://huggingface.co/docs/hub/api>

Examples

```
## Not run:
dat <- list_datasets(num_of_dataset = 50)
dat <- list_datasets(num_of_dataset = 500, chunk_size = 100)
dat <- list_datasets(author = "a4n9i")

## End(Not run)
```

mlr3hf

mlr3hf: Hugging Face datasets for mlr3

Description

Provides integration between Hugging Face datasets and the **mlr3** ecosystem. Datasets can be downloaded from the Hugging Face Hub and converted directly into [mlr3 Tasks](#).

mlr3 Integration

The package registers the "hf" task in `mlr3::mlr_tasks`.

```
task = tsk(
  "hf",
  repo_id = "scikit-learn/iris",
  config = "default",
  target = "species"
)
```

Alternatively, users can create an HFData object and convert it using `as_task()`.

Caching

Downloaded datasets are cached locally to avoid repeated downloads.

Documentation

See the package vignettes for examples of loading datasets, converting them to mlr3 tasks, and working with large datasets.

Author(s)

Maintainer: Sebastian Fischer <sebastian.fischer@stat.uni-muenchen.de>

Authors:

- Toby Hocking <toby.hocking@r-project.org>
- Anjani Nandan <anjaninandan599@gmail.com>

See Also

Useful links:

- <https://mlr-org.github.io/mlr3hf/>
- <https://github.com/mlr-org/mlr3hf>
- Report bugs at <https://github.com/mlr-org/mlr3hf/issues>

Index

`as_data_backend.HFData`, [2](#)

`as_task.HFData`, [3](#)

`cache_hfhub`, [3](#)

`cache_parquet`, [3](#), [4](#), [5](#)

`DataBackend`, [2](#)

`download_desc`, [6](#)

`fetch_dataset_meta`, [7](#)

`HFData`, [2](#), [3](#), [7](#), [10](#), [11](#)

`hfdt`, [10](#)

`list_datasets`, [11](#)

`mlr3` Tasks, [12](#)

`mlr3::as_task()`, [10](#)

`mlr3::TaskClassif`, [3](#)

`mlr3::TaskRegr`, [3](#)

`mlr3hf`, [12](#)

`mlr3hf-package (mlr3hf)`, [12](#)

`R6::R6Class`, [7](#), [8](#)

`Task`, [3](#)

`TaskClassif`, [3](#)

`TaskRegr`, [3](#)