Loading methods

Methods for listing and loading datasets:

Datasetsdatasets.load_dataset

datasets.load_datasetdatasets.load_datasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/load.py#L1190[{"name": "path", "val": ": str"}, {"name": "name", "val": ": typing.Optional[str] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, collections.abc.Sequence[str], collections.abc.Mapping[str, typing.Union[str, collections.abc.Sequence[str]]], NoneType] = None"}, {"name": "split", "val": ": typing.Union[str, datasets.splits.Split, list[str], list[datasets.splits.Split], NoneType] = None"}, {"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "download_config", "val": ": typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "download_mode", "val": ": typing.Union[datasets.download.download_manager.DownloadMode, str, NoneType] = None"}, {"name": "verification_mode", "val": ": typing.Union[datasets.utils.info_utils.VerificationMode, str, NoneType] = None"}, {"name": "keep_in_memory", "val": ": typing.Optional[bool] = None"}, {"name": "save_infos", "val": ": bool = False"}, {"name": "revision", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "streaming", "val": ": bool = False"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**config_kwargs", "val": ""}]- **path** (`str`) --
Path or name of the dataset.

- if `path` is a dataset repository on the HF hub (list all available datasets with `huggingface_hub.list_datasets`)
  -> load the dataset from supported files in the repository (csv, json, parquet, etc.)
  e.g. `'username/dataset_name'` , a dataset repository on the HF hub containing the data files.
- if `path` is a local directory
  -> load the dataset from supported files in the directory (csv, json, parquet, etc.)

e.g. `'./path/to/directory/with/my/csv/data'`.

- if `path` is the name of a dataset builder and `data_files` or `data_dir` is specified (available builders are "json", "csv", "parquet", "arrow", "text", "xml", "webdataset", "imagefolder", "audiofolder", "videofolder")
  -> load the dataset from the files in `data_files` or `data_dir`
  e.g. `'parquet'`.
- **name** (`str`, *optional*) --
  Defining the name of the dataset configuration.
- **data_dir** (`str`, *optional*) --
  Defining the `data_dir` of the dataset configuration. If specified for the generic builders (csv, text etc.) or the Hub datasets and `data_files` is `None`,
  the behavior is equal to passing `os.path.join(data_dir, **)` as `data_files` to reference all the files in a directory.
- **data_files** (`str` or `Sequence` or `Mapping`, *optional*) --
  Path(s) to source data file(s).
- **split** (`Split` or `str`) --
  Which split of the data to load.
  If `None`, will return a `dict` with all splits (typically `datasets.Split.TRAIN` and `datasets.Split.TEST`).
  If given, will return a single Dataset.
  Splits can be combined and specified like in tensorflow-datasets.
- **cache_dir** (`str`, *optional*) --
  Directory to read/write data. Defaults to `"~/.cache/huggingface/datasets"`.
- **features** (`Features`, *optional*) --
  Set the features type to use for this dataset.
- **download_config** (DownloadConfig, *optional*) --
  Specific download configuration parameters.
- **download_mode** (DownloadMode or `str`, defaults to `REUSE_DATASET_IF_EXISTS`) --
  Download/generate mode.
- **verification_mode** (VerificationMode or `str`, defaults to `BASIC_CHECKS`) --
  Verification mode determining the checks to run on the downloaded/processed dataset information (checksums/size/splits/...).
- **keep_in_memory** (`bool`, defaults to `None`) --
  Whether to copy the dataset in-memory. If `None`, the dataset
  will not be copied in-memory unless explicitly enabled by setting

`datasets.config.IN_MEMORY_MAX_SIZE` to
nonzero. See more details in the improve performance section.

- **revision** (Version or `str`, *optional*) --
  Version of the dataset to load.
  As datasets have their own git repository on the Datasets Hub, the default version "main" corresponds to their "main" branch.
  You can specify a different version than the default "main" by using a commit SHA or a git tag of the dataset repository.
- **token** (`str` or `bool`, *optional*) --
  Optional string or boolean to use as Bearer token for remote files on the Datasets Hub.
  If `True`, or not specified, will get token from `"~/.huggingface"`.
- **streaming** (`bool`, defaults to `False`) --
  If set to `True`, don't download the data files. Instead, it streams the data progressively while
  iterating on the dataset. An IterableDataset or IterableDatasetDict is returned instead in this case.
  Note that streaming works for datasets that use data formats that support being iterated over like txt, csv, jsonl for example.
  Json files may be downloaded completely. Also streaming from remote zip or gzip files is supported but other compressed formats
  like rar and xz are not yet supported. The tgz format doesn't allow streaming.
- **num_proc** (`int`, *optional*, defaults to `None`) --
  Number of processes when downloading and generating the dataset locally.
  Multiprocessing is disabled by default.
- **storage_options** (`dict`, *optional*, defaults to `None`) --
  Experimental. Key/value pairs to be passed on to the dataset file-system backend, if any.
- ****config_kwargs** (additional keyword arguments) --
  Keyword arguments to be passed to the `BuilderConfig`
  and used in the DatasetBuilder.0Dataset or DatasetDict- if `split` is not `None`: the dataset requested,
- if `split` is `None`, a DatasetDict with each split.

or IterableDataset or IterableDatasetDict: if `streaming=True`

- if `split` is not `None`, the dataset is requested
- if `split` is `None`, a `~datasets.streaming.IterableDatasetDict` with each split.

Load a dataset from the Hugging Face Hub, or a local dataset.

You can find the list of datasets on the Hub or with `huggingface_hub.list_datasets`.

A dataset is a directory that contains some data files in generic formats (JSON, CSV, Parquet, etc.) and possibly
in a generic structure (Webdataset, ImageFolder, AudioFolder, VideoFolder, etc.)

This function does the following under the hood:

1. Load a dataset builder:
   - Find the most common data format in the dataset and pick its associated builder (JSON, CSV, Parquet, Webdataset, ImageFolder, AudioFolder, etc.)
   - Find which file goes into which split (e.g. train/test) based on file and directory names or on the YAML configuration
   - It is also possible to specify `data_files` manually, and which dataset builder to use (e.g. "parquet").
2. Run the dataset builder:
   In the general case:
   - Download the data files from the dataset if they are not already available locally or cached.
   - Process and cache the dataset in typed Arrow tables for caching.
     Arrow table are arbitrarily long, typed tables which can store nested objects and be mapped to numpy/pandas/python generic types.
     They can be directly accessed from disk, loaded in RAM or even streamed over the web.

   In the streaming case:
   - Don't download or cache anything. Instead, the dataset is lazily loaded and will be streamed on-the-fly when iterating on it.
3. Return a dataset built from the requested splits in `split` (default: all).

Example:

Load a dataset from the Hugging Face Hub:

```
>>> from datasets import load_dataset
>>> ds = load_dataset('cornell-movie-review-data/rotten_tomatoes', split='train')

# Load a subset or dataset configuration (here 'sst2')
>>> from datasets import load_dataset
>>> ds = load_dataset('nyu-mll/glue', 'sst2', split='train')

# Manual mapping of data files to splits
>>> data_files = {'train': 'train.csv', 'test': 'test.csv'}
>>> ds = load_dataset('namespace/your_dataset_name', data_files=data_files)

# Manual selection of a directory to load
>>> ds = load_dataset('namespace/your_dataset_name', data_dir='folder_name')
```

Load a local dataset:

```
# Load a CSV file
>>> from datasets import load_dataset
>>> ds = load_dataset('csv', data_files='path/to/local/my_dataset.csv')

# Load a JSON file
>>> from datasets import load_dataset
>>> ds = load_dataset('json', data_files='path/to/local/my_dataset.json')
```

Load an IterableDataset:

```
>>> from datasets import load_dataset
>>> ds = load_dataset('cornell-movie-review-data/rotten_tomatoes', split='train', streaming=True)
```

Load an image dataset with the `ImageFolder` dataset builder:

```
>>> from datasets import load_dataset
>>> ds = load_dataset('imagefolder', data_dir='/path/to/images', split='train')
```

datasets.load_from_diskdatasets.load_from_diskhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/load.py#L1434[{"name": "dataset_path", "val": ": typing.Union[str, bytes, os.PathLike]"}, {"name": "keep_in_memory", "val": ": typing.Optional[bool] = None"}, {"name":

"storage_options", "val": ": typing.Optional[dict] = None"}]- **dataset_path** (`path-like`) -- Path (e.g. `"dataset/train"`) or remote URI (e.g. `"s3://my-bucket/dataset/train"`) of the Dataset or DatasetDict directory where the dataset/dataset-dict will be loaded from.

- **keep_in_memory** (`bool` , defaults to `None`) -- Whether to copy the dataset in-memory. If `None` , the dataset will not be copied in-memory unless explicitly enabled by setting `datasets.config.IN_MEMORY_MAX_SIZE` to nonzero. See more details in the improve performance section.
- **storage_options** (`dict` , *optional*) -- Key/value pairs to be passed on to the file-system backend, if any. 0Dataset or DatasetDict- If `dataset_path` is a path of a dataset directory: the dataset requested.
- If `dataset_path` is a path of a dataset dict directory, a DatasetDict with each split.

Loads a dataset that was previously saved using save_to_disk() from a dataset directory, or from a filesystem using any implementation of `fsspec.spec.AbstractFileSystem` .

Example:

```
>>> from datasets import load_from_disk
>>> ds = load_from_disk('path/to/dataset/directory')
```

datasets.load_dataset_builderdatasets.load_dataset_builderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/load.py#L1041[{"name": "path", "val": ": str"}, {"name": "name", "val": ": typing.Optional[str] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, collections.abc.Sequence[str], collections.abc.Mapping[str, typing.Union[str, collections.abc.Sequence[str]]], NoneType] = None"}, {"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "download_config", "val": ": typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "download_mode", "val": ": typing.Union[datasets.download.download_manager.DownloadMode, str, NoneType] = None"}, {"name": "revision", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name":

"storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**config_kwargs", "val": ""}]- **path** (`str`) --
Path or name of the dataset.

- if `path` is a dataset repository on the HF hub (list all available datasets with `huggingface_hub.list_datasets`)
  -> load the dataset builder from supported files in the repository (csv, json, parquet, etc.)
  e.g. `'username/dataset_name'` , a dataset repository on the HF hub containing the data files.
- if `path` is a local directory
  -> load the dataset builder from supported files in the directory (csv, json, parquet, etc.)
  e.g. `'./path/to/directory/with/my/csv/data'` .
- if `path` is the name of a dataset builder and `data_files` or `data_dir` is specified (available builders are "json", "csv", "parquet", "arrow", "text", "xml", "webdataset", "imagefolder", "audiofolder", "videofolder")
  -> load the dataset builder from the files in `data_files` or `data_dir`
  e.g. `'parquet'` .
- **name** (`str` , *optional*) --
  Defining the name of the dataset configuration.
- **data_dir** (`str` , *optional*) --
  Defining the `data_dir` of the dataset configuration. If specified for the generic builders (csv, text etc.) or the Hub datasets and `data_files` is `None` ,
  the behavior is equal to passing `os.path.join(data_dir, **)` as `data_files` to reference all the files in a directory.
- **data_files** (`str` or `Sequence` or `Mapping` , *optional*) --
  Path(s) to source data file(s).
- **cache_dir** (`str` , *optional*) --
  Directory to read/write data. Defaults to `"~/.cache/huggingface/datasets"` .
- **features** (Features, *optional*) --
  Set the features type to use for this dataset.
- **download_config** (DownloadConfig, *optional*) --
  Specific download configuration parameters.
- **download_mode** (DownloadMode or `str` , defaults to `REUSE_DATASET_IF_EXISTS`) --
  Download/generate mode.
- **revision** (Version or `str` , *optional*) --

Version of the dataset to load.
As datasets have their own git repository on the Datasets Hub, the default version "main" corresponds to their "main" branch.
You can specify a different version than the default "main" by using a commit SHA or a git tag of the dataset repository.

- **token** (`str` or `bool` , *optional*) --
  Optional string or boolean to use as Bearer token for remote files on the Datasets Hub.
  If `True` , or not specified, will get token from `"~/.huggingface"` .
- **storage_options** (`dict` , *optional*, defaults to `None`) --
  Experimental. Key/value pairs to be passed on to the dataset file-system backend, if any.
- ****config_kwargs** (additional keyword arguments) --
  Keyword arguments to be passed to the BuilderConfig
  and used in the DatasetBuilder.0DatasetBuilder
  Load a dataset builder which can be used to:
- Inspect general information that is required to build a dataset (cache directory, config, dataset info, features, data files, etc.)
- Download and prepare the dataset as Arrow files in the cache
- Get a streaming dataset without downloading or caching anything

You can find the list of datasets on the Hub or with `huggingface_hub.list_datasets` .

A dataset is a directory that contains some data files in generic formats (JSON, CSV, Parquet, etc.) and possibly
in a generic structure (Webdataset, ImageFolder, AudioFolder, VideoFolder, etc.)

Example:

```
>>> from datasets import load_dataset_builder
>>> ds_builder = load_dataset_builder('cornell-movie-review-data/rotten_tomatoes')
>>> ds_builder.info.features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
```

datasets.get_dataset_config_namesdatasets.get_dataset_config_nameshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/inspect.py#L109[{"name": "path", "val": ": str"}, {"name": "revision", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = None"}, {"name": "download_config", "val": ":

typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "download_mode", "val": ": typing.Union[datasets.download.download_manager.DownloadMode, str, NoneType] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, NoneType] = None"}, {"name": "**download_kwargs", "val": ""}]- **path** (`str`) -- path to the dataset repository. Can be either:

- a local path to the dataset directory containing the data files,
  e.g. `'./dataset/squad'`
- a dataset identifier on the Hugging Face Hub (list all available datasets and ids with `huggingface_hub.list_datasets`),
  e.g. `'rajpurkar/squad'` , `'nyu-mll/glue'` or``'openai/webtext'`
- **revision** (`Union[str, datasets.Version]` , *optional*) --
  If specified, the dataset module will be loaded from the datasets repository at this version.
  By default:
  - it is set to the local version of the lib.
  - it will also try to load it from the main branch if it's not available at the local version of the lib.
    Specifying a version that is different from your local version of the lib might cause compatibility issues.
- **download_config** (DownloadConfig, *optional*) --
  Specific download configuration parameters.
- **download_mode** (DownloadMode or `str` , defaults to `REUSE_DATASET_IF_EXISTS`) --
  Download/generate mode.
- **data_files** (`Union[Dict, List, str]` , *optional*) --
  Defining the data_files of the dataset configuration.
- ****download_kwargs** (additional keyword arguments) --
  Optional attributes for DownloadConfig which will override the attributes in `download_config` if supplied,
  for example `token` .0
  Get the list of available config names for a particular dataset.

Example:

```
>>> from datasets import get_dataset_config_names
>>> get_dataset_config_names("nyu-mll/glue")
['cola',
 'sst2',
 'mrpc',
 'qqp',
 'stsb',
 'mnli',
 'mnli_mismatched',
 'mnli_matched',
 'qnli',
 'rte',
 'wnli',
 'ax']
```

datasets.get_dataset_infosdatasets.get_dataset_infoshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/inspect.py#L42[{"name": "path", "val": ": str"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, NoneType] = None"}, {"name": "download_config", "val": ": typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "download_mode", "val": ": typing.Union[datasets.download.download_manager.DownloadMode, str, NoneType] = None"}, {"name": "revision", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "**config_kwargs", "val": ""}]- **path** (`str`) -- path to the dataset repository. Can be either:

- a local path to the dataset directory containing the data files,
  e.g. `'./dataset/squad'`
- a dataset identifier on the Hugging Face Hub (list all available datasets and ids with `huggingface_hub.list_datasets`),
  e.g. `'rajpurkar/squad'` , `'nyu-mll/glue'` or``'openai/webtext'`
- **revision** (`Union[str, datasets.Version]` , *optional*) --
  If specified, the dataset module will be loaded from the datasets repository at this version.
  By default:
  - it is set to the local version of the lib.
  - it will also try to load it from the main branch if it's not available at the local version of the lib.
    Specifying a version that is different from your local version of the lib might cause

compatibility issues.

- **download_config** (DownloadConfig, *optional*) --
  Specific download configuration parameters.
- **download_mode** (DownloadMode or `str`, defaults to `REUSE_DATASET_IF_EXISTS`) --
  Download/generate mode.
- **data_files** (`Union[Dict, List, str]`, *optional*) --
  Defining the data_files of the dataset configuration.
- **token** (`str` or `bool`, *optional*) --
  Optional string or boolean to use as Bearer token for remote files on the Datasets Hub.
  If `True`, or not specified, will get token from `"~/.huggingface"`.
- ****config_kwargs** (additional keyword arguments) --
  Optional attributes for builder class which will override the attributes if supplied.0
  Get the meta information about a dataset, returned as a dict mapping config name to
  DatasetInfoDict.

Example:

```
>>> from datasets import get_dataset_infos
>>> get_dataset_infos('cornell-movie-review-data/rotten_tomatoes')
{'default': DatasetInfo(description="Movie Review Dataset.
 is a dataset of containing 5,331 positive and 5,331 negative processed
ences from Rotten Tomatoes movie reviews...), ...}
```

datasets.get_dataset_split_namesdatasets.get_dataset_split_nameshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/inspect.py#L298[{"name": "path", "val": ": str"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, collections.abc.Sequence[str], collections.abc.Mapping[str, typing.Union[str, collections.abc.Sequence[str]]], NoneType] = None"}, {"name": "download_config", "val": ": typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "download_mode", "val": ": typing.Union[datasets.download.download_manager.DownloadMode, str, NoneType] = None"}, {"name": "revision", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "**config_kwargs", "val": ""}]- **path** (`str`) -- path to the dataset repository. Can be either:

- a local path to the dataset directory containing the data files,

e.g. `'./dataset/squad'`

- a dataset identifier on the Hugging Face Hub (list all available datasets and ids with `huggingface_hub.list_datasets`),
  e.g. `'rajpurkar/squad'`, `'nyu-mll/glue'` or``'openai/webtext'`
- **config_name** (`str`, *optional*) --
  Defining the name of the dataset configuration.
- **data_files** (`str` or `Sequence` or `Mapping`, *optional*) --
  Path(s) to source data file(s).
- **download_config** (DownloadConfig, *optional*) --
  Specific download configuration parameters.
- **download_mode** (DownloadMode or `str`, defaults to `REUSE_DATASET_IF_EXISTS`) --
  Download/generate mode.
- **revision** (Version or `str`, *optional*) --
  Version of the dataset to load.
  As datasets have their own git repository on the Datasets Hub, the default version "main" corresponds to their "main" branch.
  You can specify a different version than the default "main" by using a commit SHA or a git tag of the dataset repository.
- **token** (`str` or `bool`, *optional*) --
  Optional string or boolean to use as Bearer token for remote files on the Datasets Hub.
  If `True`, or not specified, will get token from `"~/.huggingface"`.
- ****config_kwargs** (additional keyword arguments) --
  Optional attributes for builder class which will override the attributes if supplied.0
  Get the list of available splits for a particular config and dataset.

Example:

```
>>> from datasets import get_dataset_split_names
>>> get_dataset_split_names('cornell-movie-review-data/rotten_tomatoes')
['train', 'validation', 'test']
```

From files

Configurations used to load data files.
They are used when loading local files or a dataset repository:

- local files: `load_dataset("parquet", data_dir="path/to/data/dir")`
- dataset repository: `load_dataset("allenai/c4")`

You can pass arguments to `load_dataset` to configure data loading.
For example you can specify the `sep` parameter to define the CsvConfig that is used to load the data:

```
load_dataset("csv", data_dir="path/to/data/dir", sep="\t")
```

Textdatasets.packaged_modules.text.TextConfig

class datasets.packaged_modules.text.TextConfigdatasets.packaged_modules.text.TextConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/text/text.py#L17[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "encoding", "val": ": str = 'utf-8'"}, {"name": "encoding_errors", "val": ": typing.Optional[str] = None"}, {"name": "chunksize", "val": ": int = 10485760"}, {"name": "keep_linebreaks", "val": ": bool = False"}, {"name": "sample_by", "val": ": str = 'line'"}]
BuilderConfig for text files.

class datasets.packaged_modules.text.Textdatasets.packaged_modules.text.Texthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/text/text.py#L28[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ":

typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

CSVdatasets.packaged_modules.csv.CsvConfig

class datasets.packaged_modules.csv.CsvConfigdatasets.packaged_modules.csv.CsvConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/csv/csv.py#L25[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "sep", "val": ": str = ','"}, {"name": "delimiter", "val": ": typing.Optional[str] = None"}, {"name": "header", "val": ": typing.Union[int, list[int], str, NoneType] = 'infer'"}, {"name": "names", "val": ": typing.Optional[list[str]] = None"}, {"name": "column_names", "val": ": typing.Optional[list[str]] = None"}, {"name": "index_col", "val": ": typing.Union[int, str, list[int], list[str], NoneType] = None"}, {"name": "usecols", "val": ": typing.Union[list[int], list[str], NoneType] = None"}, {"name": "prefix", "val": ": typing.Optional[str] = None"}, {"name": "mangle_dupe_cols", "val": ": bool = True"}, {"name": "engine", "val": ": typing.Optional[typing.Literal['c', 'python', 'pyarrow']] = None"}, {"name": "converters", "val": ": dict = None"}, {"name": "true_values", "val": ": typing.Optional[list] = None"}, {"name": "false_values", "val": ": typing.Optional[list] = None"}, {"name": "skipinitialspace", "val": ": bool = False"}, {"name": "skiprows", "val": ": typing.Union[int, list[int], NoneType] = None"}, {"name": "nrows", "val": ": typing.Optional[int] = None"}, {"name": "na_values", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "keep_default_na", "val": ": bool = True"}, {"name": "na_filter", "val": ": bool = True"}, {"name": "verbose", "val": ": bool = False"}, {"name": "skip_blank_lines", "val": ": bool = True"}, {"name": "thousands", "val": ": typing.Optional[str] = None"}, {"name": "decimal", "val": ": str = '.'"}, {"name": "lineterminator", "val": ": typing.Optional[str] = None"}, {"name": "quotechar", "val": ": str = '"'"}, {"name": "quoting", "val": ": int = 0"}, {"name": "escapechar", "val": ": typing.Optional[str] = None"}, {"name": "comment", "val": ": typing.Optional[str] = None"}, {"name": "encoding", "val": ": typing.Optional[str] = None"}, {"name": "dialect", "val": ": typing.Optional[str] = None"}, {"name": "error_bad_lines", "val": ": bool = True"}, {"name": "warn_bad_lines", "val": ": bool = True"}, {"name": "skipfooter", "val": ": int = 0"}, {"name":

"doublequote", "val": ": bool = True"}, {"name": "memory_map", "val": ": bool = False"}, {"name": "float_precision", "val": ": typing.Optional[str] = None"}, {"name": "chunksize", "val": ": int = 10000"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "encoding_errors", "val": ": typing.Optional[str] = 'strict'"}, {"name": "on_bad_lines", "val": ": typing.Literal['error', 'warn', 'skip'] = 'error'"}, {"name": "date_format", "val": ": typing.Optional[str] = None"}]
BuilderConfig for CSV.

class datasets.packaged_modules.csv.Csvdatasets.packaged_modules.csv.Csvhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/csv/csv.py#L145[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

JSONdatasets.packaged_modules.json.JsonConfig

class datasets.packaged_modules.json.JsonConfigdatasets.packaged_modules.json.JsonConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/json/json.py#L42[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "encoding", "val": ": str = 'utf-8'"}, {"name": "encoding_errors", "val": ": typing.Optional[str] = None"}, {"name": "field", "val": ": typing.Optional[str] = None"}, {"name": "use_threads", "val": ":

bool = True"}, {"name": "block_size", "val": ": typing.Optional[int] = None"}, {"name": "chunksize", "val": ": int = 10485760"}, {"name": "newlines_in_values", "val": ": typing.Optional[bool] = None"}]
BuilderConfig for JSON.

class datasets.packaged_modules.json.Jsondatasets.packaged_modules.json.Jsonhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/json/json.py#L58[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

XMLdatasets.packaged_modules.xml.XmlConfig

class datasets.packaged_modules.xml.XmlConfigdatasets.packaged_modules.xml.XmlConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/xml/xml.py#L16[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "encoding", "val": ": str = 'utf-8'"}, {"name": "encoding_errors", "val": ": typing.Optional[str] = None"}]
BuilderConfig for xml files.

class

datasets.packaged_modules.xml.Xmldatasets.packaged_modules.xml.Xmlhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/xml/xml.py#L24[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

Parquetdatasets.packaged_modules.parquet.ParquetConfig

class datasets.packaged_modules.parquet.ParquetConfigdatasets.packaged_modules.parquet.ParquetConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/parquet/parquet.py#L17[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "columns", "val": ": typing.Optional[list[str]] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "filters", "val": ": typing.Union[pyarrow._compute.Expression, list[tuple], list[list[tuple]], NoneType] = None"}, {"name": "fragment_scan_options", "val": ": typing.Optional[pyarrow._dataset_parquet.ParquetFragmentScanOptions] = None"}, {"name": "on_bad_files", "val": ": typing.Literal['error', 'warn', 'skip'] = 'error'"}]- **batch_size** (`int`, *optional*) --
Size of the RecordBatches to iterate on.
The default is the row group size (defined by the first row group).

- **columns** (`list[str]` , *optional*) --
  List of columns to load, the other ones are ignored.
  All columns are loaded by default.
- **features** -- (`Features` , *optional*):
  Cast the data to `features` .
- **filters** (`Union[pyarrow.dataset.Expression, list[tuple], list[list[tuple]]]` , *optional*) --
  Return only the rows matching the filter.
  If possible the predicate will be pushed down to exploit the partition information
  or internal metadata found in the data source, e.g. Parquet statistics.
  Otherwise filters the loaded RecordBatches before yielding them.
- **fragment_scan_options** (`pyarrow.dataset.ParquetFragmentScanOptions` , *optional*) --
  Scan-specific options for Parquet fragments.
  This is especially useful to configure buffering and caching.
- **on_bad_files** (`Literal["error", "warn", "skip"]` , *optional*, defaults to "error") --
  Specify what to do upon encountering a bad file (a file that can't be read). Allowed values are :
  - 'error', raise an Exception when a bad file is encountered.
  - 'warn', raise a warning when a bad file is encountered and skip that file.
  - 'skip', skip bad files without raising or warning when they are encountered.

  0

BuilderConfig for Parquet.

Example:

Load a subset of columns:

```
>>> ds = load_dataset(parquet_dataset_id, columns=["col_0", "col_1"])
```

Stream data and efficiently filter data, possibly skipping entire files or row groups:

```
>>> filters = [("col_0", "==", 0)]
>>> ds = load_dataset(parquet_dataset_id, streaming=True, filters=filters)
```

Increase the minimum request size when streaming from 32MiB (default) to 128MiB and enable prefetching:

```
>>> import pyarrow
>>> import pyarrow.dataset
>>> fragment_scan_options = pyarrow.dataset.ParquetFragmentScanOptions(
...     cache_options=pyarrow.CacheOptions(
...         prefetch_limit=1,
...         range_size_limit=128 << 20
...     ),
... )
>>> ds = load_dataset(parquet_dataset_id, streaming=True, fragment_scan_options=fragment_scan_opti
```

class datasets.packaged_modules.parquet.Parquetdatasets.packaged_modules.parquet.Parquethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/parquet/parquet.py#L90[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

Arrowdatasets.packaged_modules.arrow.ArrowConfig

class datasets.packaged_modules.arrow.ArrowConfigdatasets.packaged_modules.arrow.ArrowConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/arrow/arrow.py#L15[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name":

"features", "val": ": typing.Optional[datasets.features.features.Features] = None"}]
BuilderConfig for Arrow.

class datasets.packaged_modules.arrow.Arrowdatasets.packaged_modules.arrow.Arrowhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/arrow/arrow.py#L24[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

SQLdatasets.packaged_modules.sql.SqlConfig

class datasets.packaged_modules.sql.SqlConfigdatasets.packaged_modules.sql.SqlConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/sql/sql.py#L24[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "sql", "val": ": typing.Union[str, ForwardRef('sqlalchemy.sql.Selectable')] = None"}, {"name": "con", "val": ": typing.Union[str, ForwardRef('sqlalchemy.engine.Connection'), ForwardRef('sqlalchemy.engine.Engine'), ForwardRef('sqlite3.Connection')] = None"}, {"name": "index_col", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "coerce_float", "val": ": bool = True"}, {"name": "params", "val": ": typing.Union[list, tuple, dict, NoneType] = None"}, {"name": "parse_dates", "val": ": typing.Union[list, dict, NoneType] = None"}, {"name": "columns", "val": ": typing.Optional[list[str]] = None"}, {"name": "chunksize", "val": ":

typing.Optional[int] = 10000"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}] BuilderConfig for SQL.

class datasets.packaged_modules.sql.Sqldatasets.packaged_modules.sql.Sqlhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/sql/sql.py#L91[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

Imagesdatasets.packaged_modules.imagefolder.ImageFolderConfig

class datasets.packaged_modules.imagefolder.ImageFolderConfigdatasets.packaged_modules.imagefolder.ImageFolderConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/imagefolder/imagefolder.py#L9[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "drop_labels", "val": ": bool = None"}, {"name": "drop_metadata", "val": ": bool = None"}, {"name": "metadata_filenames", "val": ": list = None"}, {"name": "filters", "val": ": typing.Union[pyarrow._compute.Expression, list[tuple], list[list[tuple]], NoneType] = None"}] BuilderConfig for ImageFolder.

class datasets.packaged_modules.imagefolder.ImageFolderdatasets.packaged_modules.imagefolder.ImageFolderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/imagefolder/imagefolder.py#L19[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

Audiodatasets.packaged_modules.audiofolder.AudioFolderConfig

class datasets.packaged_modules.audiofolder.AudioFolderConfigdatasets.packaged_modules.audiofolder.AudioFolderConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/audiofolder/audiofolder.py#L9[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "drop_labels", "val": ": bool = None"}, {"name": "drop_metadata", "val": ": bool = None"}, {"name": "metadata_filenames", "val": ": list = None"}, {"name": "filters", "val": ": typing.Union[pyarrow._compute.Expression, list[tuple], list[list[tuple]], NoneType] = None"}]
Builder Config for AudioFolder.

class datasets.packaged_modules.audiofolder.AudioFolderdatasets.packaged_modules.audiofolder.AudioFolderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/

packaged_modules/audiofolder/audiofolder.py#L19[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

Videosdatasets.packaged_modules.videofolder.VideoFolderConfig

class datasets.packaged_modules.videofolder.VideoFolderConfigdatasets.packaged_modules.videofolder.VideoFolderConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/videofolder/videofolder.py#L9[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "drop_labels", "val": ": bool = None"}, {"name": "drop_metadata", "val": ": bool = None"}, {"name": "metadata_filenames", "val": ": list = None"}, {"name": "filters", "val": ": typing.Union[pyarrow._compute.Expression, list[tuple], list[list[tuple]], NoneType] = None"}]
BuilderConfig for ImageFolder.

class datasets.packaged_modules.videofolder.VideoFolderdatasets.packaged_modules.videofolder.VideoFolderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/videofolder/videofolder.py#L19[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ":

typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

HDF5datasets.packaged_modules.hdf5.HDF5Config

class datasets.packaged_modules.hdf5.HDF5Configdatasets.packaged_modules.hdf5.HDF5Config https://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/hdf5/hdf5.py#L33[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}]
BuilderConfig for HDF5.

class datasets.packaged_modules.hdf5.HDF5datasets.packaged_modules.hdf5.HDF5https://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/hdf5/hdf5.py#L40[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ":

typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]
ArrowBasedBuilder that converts HDF5 files to Arrow tables using the HF extension types.

Pdfdatasets.packaged_modules.pdffolder.PdfFolderConfig

class datasets.packaged_modules.pdffolder.PdfFolderConfigdatasets.packaged_modules.pdffolder.PdfFolderConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/pdffolder/pdffolder.py#L9[{"name": "name", "val": ": str = 'default'"}, {"name": "version", "val": ": typing.Union[datasets.utils.version.Version, str, NoneType] = 0.0.0"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[datasets.data_files.DataFilesDict, datasets.data_files.DataFilesPatternsDict, NoneType] = None"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "drop_labels", "val": ": bool = None"}, {"name": "drop_metadata", "val": ": bool = None"}, {"name": "metadata_filenames", "val": ": list = None"}, {"name": "filters", "val": ": typing.Union[pyarrow._compute.Expression, list[tuple], list[list[tuple]], NoneType] = None"}]
BuilderConfig for ImageFolder.

class datasets.packaged_modules.pdffolder.PdfFolderdatasets.packaged_modules.pdffolder.PdfFolderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/pdffolder/pdffolder.py#L19[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]

WebDatasetdatasets.packaged_modules.webdataset.WebDataset

class datasets.packaged_modules.webdataset.WebDatasetdatasets.packaged_modules.webdataset.WebDatasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/packaged_modules/webdataset/webdataset.py#L19[{"name": "cache_dir", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "hash", "val": ": typing.Optional[str] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "token", "val": ": typing.Union[bool, str, NoneType] = None"}, {"name": "repo_id", "val": ": typing.Optional[str] = None"}, {"name": "data_files", "val": ": typing.Union[str, list, dict, datasets.data_files.DataFilesDict, NoneType] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**config_kwargs", "val": ""}]