# Builder classes

## Buildersdatasets.DatasetBuilder

Datasets relies on two main classes during the dataset building process: DatasetBuilder and BuilderConfig.

class datasets.DatasetBuilderdatasets.DatasetBuilderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L210[{"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": ""}]- **cache_dir** ( `str` , *optional*) --
Directory to cache data. Defaults to `"~/.cache/huggingface/datasets"` .

- **dataset_name** ( `str` , *optional*) --
  Name of the dataset, if different from the builder name. Useful for packaged builders like csv, imagefolder, audiofolder, etc. to reflect the difference between datasets that use the same packaged builder.
- **config_name** ( `str` , *optional*) --
  Name of the dataset configuration.
  It affects the data generated on disk. Different configurations will have their own subdirectories and
  versions.
  If not provided, the default configuration is used (if it exists).
  Parameter `name` was renamed to `config_name` .
- **hash** ( `str` , *optional*) --

Hash specific to the dataset builder code. Used to update the caching directory when the dataset builder code is updated (to avoid reusing old data).
The typical caching directory (defined in `self._relative_data_dir`) is `name/version/hash/`.

- **base_path** (`str`, *optional*) --
  Base path for relative paths that are used to download files.
  This can be a remote URL.
- **features** (Features, *optional*) --
  Features types to use with this dataset.
  It can be used to change the Features types of a dataset, for example.
- **token** (`str` or `bool`, *optional*) --
  String or boolean to use as Bearer token for remote files on the
  Datasets Hub. If `True`, will get token from `"~/.huggingface"`.
- **repo_id** (`str`, *optional*) --
  ID of the dataset repository.
  Used to distinguish builders with the same name but not coming from the same namespace, for example "rajpurkar/squad"
  and "lhoestq/squad" repo IDs. In the latter, the builder name would be "lhoestq___squad".
- **data_files** (`str` or `Sequence` or `Mapping`, *optional*) --
  Path(s) to source data file(s).
  For builders like "csv" or "json" that need the user to specify data files. They can be either local or remote files. For convenience, you can use a `DataFilesDict`.
- **data_dir** (`str`, *optional*) --
  Path to directory containing source data file(s).
  Use only if `data_files` is not passed, in which case it is equivalent to passing `os.path.join(data_dir, "**")` as `data_files`.
  For builders that require manual download, it must be the path to the local directory containing the
  manually downloaded data.
- **storage_options** (`dict`, *optional*) --
  Key/value pairs to be passed on to the dataset file-system backend, if any.
- **writer_batch_size** (`int`, *optional*) --
  Batch size used by the ArrowWriter.
  It defines the number of samples that are kept in memory before writing them and also the length of the arrow chunks.

None means that the ArrowWriter will use its default value.

- ****config_kwargs** (additional keyword arguments) -- Keyword arguments to be passed to the corresponding builder
  configuration class, set on the class attribute DatasetBuilder.BUILDER_CONFIG_CLASS. The builder
  configuration class is BuilderConfig or a subclass of it.0
  Abstract base class for all datasets.

`DatasetBuilder` has 3 key methods:

- `DatasetBuilder.info`: Documents the dataset, including feature names, types, shapes, version, splits, citation, etc.
- DatasetBuilder.download_and_prepare(): Downloads the source data and writes it to disk.
- DatasetBuilder.as_dataset(): Generates a Dataset.

Some `DatasetBuilder`s expose multiple variants of the dataset by defining a BuilderConfig subclass and accepting a config object (or name) on construction. Configurable datasets expose a pre-defined set of configurations in `DatasetBuilder.builder_configs()`.

as_datasetdatasets.DatasetBuilder.as_datasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L1027[{"name": "split", "val": ": typing.Union[str, datasets.splits.Split, list[str], list[datasets.splits.Split], NoneType] = None"}, {"name": "run_post_process", "val": " = True"}, {"name": "verification_mode", "val": ": typing.Union[datasets.utils.info_utils.VerificationMode, str, NoneType] = None"}, {"name": "in_memory", "val": " = False"}]- **split** ( `datasets.Split` ) --
Which subset of the data to return.

- **run_post_process** ( `bool` , defaults to `True` ) --
  Whether to run post-processing dataset transforms and/or add indexes.
- **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/...).
- **in_memory** ( `bool` , defaults to `False` ) --
  Whether to copy the data in-memory.0datasets.Dataset

Return a Dataset for the specified split.

Example:

```
>>> from datasets import load_dataset_builder
>>> builder = load_dataset_builder('cornell-movie-review-data/rotten_tomatoes')
>>> builder.download_and_prepare()
>>> ds = builder.as_dataset(split='train')
>>> ds
Dataset({
    features: ['text', 'label'],
    num_rows: 8530
})
```

download_and_preparedatasets.DatasetBuilder.download_and_preparehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L691[{"name": "output_dir", "val": ": typing.Optional[str] = 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": "dl_manager", "val": ": typing.Optional[datasets.download.download_manager.DownloadManager] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "file_format", "val": ": str = 'arrow'"}, {"name": "max_shard_size", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**download_and_prepare_kwargs", "val": ""}]-
**output_dir** ( `str` , *optional*) --
Output directory for the dataset.
Default to this builder's `cache_dir` , which is inside `~/.cache/huggingface/datasets` by default.

- **download_config** (`DownloadConfig`, *optional*) -- Specific download configuration parameters. - **download_mode** ([DownloadMode](/docs/datasets/v4.2.0/en/package_reference/builder_classes#datasets.DownloadMode) or `str`, *optional*) -- Select the download/generate mode, default to `REUSE_DATASET_IF_EXISTS`. - **verification_mode** ([VerificationMode](/docs/datasets/v4.2.0/en/package_reference/builder_classes#datasets.VerificationMode) or `str`, defaults to `BASIC_CHECKS`) --

Verification mode determining the checks to run on the downloaded/processed dataset information (checksums/size/splits/...). - **dl_manager** (`DownloadManager`, *optional*) -- Specific `DownloadManger` to use. - **base_path** (`str`, *optional*) -- Base path for relative paths that are used to download files. This can be a remote url. If not specified, the value of the `base_path` attribute (`self.base_path`) will be used instead. - **file_format** (`str`, *optional*) -- Format of the data files in which the dataset will be written. Supported formats: "arrow", "parquet". Default to "arrow" format. If the format is "parquet", then image and audio data are embedded into the Parquet files instead of pointing to local files. - **max_shard_size** (`Union[str, int]`, *optional*) -- Maximum number of bytes written per shard, default is "500MB". The size is based on uncompressed data size, so in practice your shard files may be smaller than `max_shard_size` thanks to Parquet compression for example. - **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*) -- Key/value pairs to be passed on to the caching file-system backend, if any. - ****download_and_prepare_kwargs** (additional keyword arguments) -- Keyword arguments.0
Downloads and prepares dataset for reading.
Example:

Download and prepare the dataset as Arrow files that can be loaded as a Dataset using `builder.as_dataset()`:

```
>>> from datasets import load_dataset_builder
>>> builder = load_dataset_builder("cornell-movie-review-data/rotten_tomatoes")
>>> builder.download_and_prepare()
```

Download and prepare the dataset as sharded Parquet files locally:

```
>>> from datasets import load_dataset_builder
>>> builder = load_dataset_builder("cornell-movie-review-data/rotten_tomatoes")
>>> builder.download_and_prepare("./output_dir", file_format="parquet")
```

Download and prepare the dataset as sharded Parquet files in a cloud storage:

```
>>> from datasets import load_dataset_builder
>>> storage_options = {"key": aws_access_key_id, "secret": aws_secret_access_key}
>>> builder = load_dataset_builder("cornell-movie-review-data/rotten_tomatoes")
>>> builder.download_and_prepare("s3://my-bucket/my_rotten_tomatoes", storage_options=storage_opti
```

get_imported_module_dirdatasets.DatasetBuilder.get_imported_module_dirhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L683[]
Return the path of the module of this class or subclass.

class datasets.GeneratorBasedBuilderdatasets.GeneratorBasedBuilderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L1356[{"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": ""}]
Base class for datasets with data generation based on dict generators.

`GeneratorBasedBuilder` is a convenience class that abstracts away much of the data writing and reading of `DatasetBuilder`. It expects subclasses to implement generators of feature dictionaries across the dataset splits (`_split_generators`). See the method docstrings for details.

class datasets.ArrowBasedBuilderdatasets.ArrowBasedBuilderhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L1621[{"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": ""}]
Base class for datasets with data generation based on Arrow loading functions (CSV/JSON/Parquet).

class datasets.BuilderConfigdatasets.BuilderConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L97[{"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** ( `str`, defaults to `default` ) --
The name of the configuration.

- **version** ( `Version` or `str`, defaults to `0.0.0` ) --
  The version of the configuration.
- **data_dir** ( `str`, *optional*) --
  Path to the directory containing the source data.
- **data_files** ( `str` or `Sequence` or `Mapping`, *optional*) --
  Path(s) to source data file(s).
- **description** ( `str`, *optional*) --
  A human description of the configuration.0
  Base class for `DatasetBuilder` data configuration.

`DatasetBuilder` subclasses with data configuration options should subclass `BuilderConfig` and add their own properties.

create_config_iddatasets.BuilderConfig.create_config_idhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/builder.py#L140[{"name": "config_kwargs", "val": ": dict"}, {"name": "custom_features", "val": ": typing.Optional[datasets.features.features.Features] = None"}]

The config id is used to build the cache directory.

By default it is equal to the config name.
However the name of a config is not sufficient to have a unique identifier for the dataset being generated
since it doesn't take into account:

- the config kwargs that can be used to overwrite attributes
- the custom features used to write the dataset
- the data_files for json/text/csv/pandas datasets

Therefore the config id is just the config name with an optional suffix based on these.

# Downloaddatasets.DownloadManager

class datasets.DownloadManagerdatasets.DownloadManagerhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L71[{"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "download_config", "val": ": typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}, {"name": "record_checksums", "val": " = True"}]

downloaddatasets.DownloadManager.downloadhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L131[{"name": "url_or_urls", "val": ""}]- **url_or_urls** ( `str` or `list` or `dict` ) --
URL or `list` or `dict` of URLs to download. Each URL is a `str` .0 `str` or `list` or `dict` The downloaded paths matching the given input `url_or_urls` .
Download given URL(s).

By default, only one process is used for download. Pass customized `download_config.num_proc` to change this behavior.

Example:

```
>>> downloaded_files = dl_manager.download('https://storage.googleapis.com/seldon-datasets/sentenc
```

download_and_extractdatasets.DownloadManager.download_and_extracthttps://github.com/

huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L310[{"name": "url_or_urls", "val": ""}]- **url_or_urls** ( `str` or `list` or `dict` ) --
URL or `list` or `dict` of URLs to download and extract. Each URL is a `str` .0extracted_path(s) `str` , extracted paths of given URL(s).
Download and extract given `url_or_urls` .

Is roughly equivalent to:

```
extracted_paths = dl_manager.extract(dl_manager.download(url_or_urls))
```

extractdatasets.DownloadManager.extracthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L278[{"name": "path_or_paths", "val": ""}]- **path_or_paths** (path or `list` or `dict` ) --
Path of file to extract. Each path is a `str` .0extracted_path(s) `str` , The extracted paths matching the given input
path_or_paths.
Extract given path(s).

Example:

```
>>> downloaded_files = dl_manager.download('https://storage.googleapis.com/seldon-datasets/sentenc
>>> extracted_files = dl_manager.extract(downloaded_files)
```

iter_archivedatasets.DownloadManager.iter_archivehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L234[{"name": "path_or_buf", "val": ": typing.Union[str, _io.BufferedReader]"}]- **path_or_buf** ( `str` or `io.BufferedReader` ) --
Archive path or archive binary file object.0 `tuple[str, io.BufferedReader]` 2-tuple (path_within_archive, file_object).
File object is opened in binary mode.
Iterate over files within an archive.

Example:

```
>>> archive = dl_manager.download('https://storage.googleapis.com/seldon-datasets/sentence_polarit
>>> files = dl_manager.iter_archive(archive)
```

iter_filesdatasets.DownloadManager.iter_fileshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L259[{"name": "paths", "val": ": typing.Union[str, list[str]]"}]- **paths** ( `str` or `list` of `str` ) --
Root paths.0 `str` File path.
Iterate over file paths.

Example:

```
>>> files = dl_manager.download_and_extract('https://huggingface.co/datasets/beans/resolve/main/da
>>> files = dl_manager.iter_files(files)
```

class datasets.StreamingDownloadManagerdatasets.StreamingDownloadManagerhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/streaming_download_manager.py#L47[{"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "download_config", "val": ": typing.Optional[datasets.download.download_config.DownloadConfig] = None"}, {"name": "base_path", "val": ": typing.Optional[str] = None"}]

Download manager that uses the "::" separator to navigate through (possibly remote) compressed archives.
Contrary to the regular `DownloadManager`, the `download` and `extract` methods don't actually download nor extract
data, but they rather return the path or url that could be opened using the `xopen` function which extends the
built-in `open` function to stream data from remote files.

downloaddatasets.StreamingDownloadManager.downloadhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/streaming_download_manager.py#L75[{"name": "url_or_urls", "val": ""}]- **url_or_urls** ( `str` or `list` or `dict` ) --
URL(s) of files to stream data from. Each url is a `str` .0url(s)( `str` or `list` or `dict` ), URL(s) to stream data from matching the given input url_or_urls.
Normalize URL(s) of files to stream data from.
This is the lazy version of `DownloadManager.download` for streaming.

Example:

```
>>> downloaded_files = dl_manager.download('https://storage.googleapis.com/seldon-datasets/sentenc
```

download_and_extractdatasets.StreamingDownloadManager.download_and_extracthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/streaming_download_manager.py#L151[{"name": "url_or_urls", "val": ""}]- **url_or_urls** ( `str` or `list` or `dict` ) --
URL(s) to stream from data from. Each url is a `str` .0url(s)( `str` or `list` or `dict` ), URL(s) to stream data from matching the given input `url_or_urls` .
Prepare given `url_or_urls` for streaming (add extraction protocol).

This is the lazy version of `DownloadManager.download_and_extract` for streaming.

Is equivalent to:

```
urls = dl_manager.extract(dl_manager.download(url_or_urls))
```

extractdatasets.StreamingDownloadManager.extracthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/streaming_download_manager.py#L102[{"name": "url_or_urls", "val": ""}]- **url_or_urls** ( `str` or `list` or `dict` ) --
URL(s) of files to stream data from. Each url is a `str` .0url(s)( `str` or `list` or `dict` ), URL(s) to stream data from matching the given input `url_or_urls` .
Add extraction protocol for given url(s) for streaming.

This is the lazy version of `DownloadManager.extract` for streaming.

Example:

```
>>> downloaded_files = dl_manager.download('https://storage.googleapis.com/seldon-datasets/sentenc
>>> extracted_files = dl_manager.extract(downloaded_files)
```

iter_archivedatasets.StreamingDownloadManager.iter_archivehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/streaming_download_manager.py#L171[{"name": "urlpath_or_buf", "val": ": typing.Union[str, _io.BufferedReader]"}]- **urlpath_or_buf** ( `str` or `io.BufferedReader` ) --
Archive path or archive binary file object.0 `tuple[str, io.BufferedReader]` 2-tuple (path_within_archive, file_object).

File object is opened in binary mode.
Iterate over files within an archive.

Example:

```
>>> archive = dl_manager.download('https://storage.googleapis.com/seldon-datasets/sentence_polarit
>>> files = dl_manager.iter_archive(archive)
```

iter_filesdatasets.StreamingDownloadManager.iter_fileshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/streaming_download_manager.py#L196[{"name": "urlpaths", "val": ": typing.Union[str, list[str]]"}]- **urlpaths** ( `str` or `list` of `str` ) --
Root paths.0strFile URL path.
Iterate over files.

Example:

```
>>> files = dl_manager.download_and_extract('https://huggingface.co/datasets/beans/resolve/main/da
>>> files = dl_manager.iter_files(files)
```

class datasets.DownloadConfigdatasets.DownloadConfighttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_config.py#L10[{"name": "cache_dir", "val": ": typing.Union[str, pathlib.Path, NoneType] = None"}, {"name": "force_download", "val": ": bool = False"}, {"name": "resume_download", "val": ": bool = False"}, {"name": "local_files_only", "val": ": bool = False"}, {"name": "proxies", "val": ": typing.Optional[dict] = None"}, {"name": "user_agent", "val": ": typing.Optional[str] = None"}, {"name": "extract_compressed_file", "val": ": bool = False"}, {"name": "force_extract", "val": ": bool = False"}, {"name": "delete_extracted", "val": ": bool = False"}, {"name": "extract_on_the_fly", "val": ": bool = False"}, {"name": "use_etag", "val": ": bool = True"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "max_retries", "val": ": int = 1"}, {"name": "token", "val": ": typing.Union[str, bool, NoneType] = None"}, {"name": "storage_options", "val": ": dict = "}, {"name": "download_desc", "val": ": typing.Optional[str] = None"}, {"name": "disable_tqdm", "val": ": bool = False"}]- **cache_dir** ( `str` or `Path` , *optional*) --
Specify a cache directory to save the file to (overwrite the
default cache dir).

- **force_download** ( `bool` , defaults to `False` ) --

If `True`, re-download the file even if it's already cached in the cache dir.

- **resume_download** (`bool`, defaults to `False`) --
  If `True`, resume the download if an incompletely received file is found.
- **proxies** (`dict`, *optional*) --
- **user_agent** (`str`, *optional*) --
  Optional string or dict that will be appended to the user-agent on remote requests.
- **extract_compressed_file** (`bool`, defaults to `False`) --
  If `True` and the path point to a zip or tar file, extract the compressed file in a folder along the archive.
- **force_extract** (`bool`, defaults to `False`) --
  If `True` when `extract_compressed_file` is `True` and the archive was already extracted, re-extract the archive and override the folder where it was extracted.
- **delete_extracted** (`bool`, defaults to `False`) --
  Whether to delete (or keep) the extracted files.
- **extract_on_the_fly** (`bool`, defaults to `False`) --
  If `True`, extract compressed files while they are being read.
- **use_etag** (`bool`, defaults to `True`) --
  Whether to use the ETag HTTP response header to validate the cached files.
- **num_proc** (`int`, *optional*) --
  The number of processes to launch to download the files in parallel.
- **max_retries** (`int`, default to `1`) --
  The number of times to retry an HTTP request if it fails.
- **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*) --
  Key/value pairs to be passed on to the dataset file-system backend, if any.
- **download_desc** (`str`, *optional*) --
  A description to be displayed alongside with the progress bar while downloading the files.
- **disable_tqdm** (`bool`, defaults to `False`) --

Whether to disable the individual files download progress bar0
Configuration for our cached path manager.

class datasets.DownloadModedatasets.DownloadModehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/download/download_manager.py#L50[{"name": "value", "val": ""}, {"name": "names", "val": " = None"}, {"name": "module", "val": " = None"}, {"name": "qualname", "val": " = None"}, {"name": "type", "val": " = None"}, {"name": "start", "val": " = 1"}]
`Enum` for how to treat pre-existing downloads and data.

The default mode is `REUSE_DATASET_IF_EXISTS`, which will reuse both
raw downloads and the prepared dataset if they exist.

The generations modes:

| | Downloads | Dataset |
|---|---|---|
| `REUSE_DATASET_IF_EXISTS` (default) | Reuse | Reuse |
| `REUSE_CACHE_IF_EXISTS` | Reuse | Fresh |
| `FORCE_REDOWNLOAD` | Fresh | Fresh |

# Verificationdatasets.VerificationMode

class datasets.VerificationModedatasets.VerificationModehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/utils/info_utils.py#L22[{"name": "value", "val": ""}, {"name": "names", "val": " = None"}, {"name": "module", "val": " = None"}, {"name": "qualname", "val": " = None"}, {"name": "type", "val": " = None"}, {"name": "start", "val": " = 1"}]
`Enum` that specifies which verification checks to run.

The default mode is `BASIC_CHECKS`, which will perform only rudimentary checks to avoid slowdowns
when generating/downloading a dataset for the first time.

The verification modes:

| | Verification checks |
|---|---|
| `ALL_CHECKS` | Split checks, uniqueness of the keys yielded in case of the GeneratorBuilder |
| | and the validity (number of files, checksums, etc.) of downloaded files |
| `BASIC_CHECKS` (default) | Same as `ALL_CHECKS` but without checking downloaded files |
| `NO_CHECKS` | None |

# Splitsdatasets.SplitGenerator

class datasets.SplitGeneratordatasets.SplitGeneratorhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/splits.py#L602[{"name": "name", "val": ": str"}, {"name": "gen_kwargs", "val": ": dict = "}]- **name** ( `str` ) --
Name of the `Split` for which the generator will
create the examples.

- ****gen_kwargs** (additional keyword arguments) --
  Keyword arguments to forward to the `DatasetBuilder._generate_examples` method of the builder.0
  Defines the split information for the generator.

This should be used as returned value of
`GeneratorBasedBuilder._split_generators` .
See `GeneratorBasedBuilder._split_generators` for more info and example
of usage.

Example:

```
>>> datasets.SplitGenerator(
...     name=datasets.Split.TRAIN,
...     gen_kwargs={"split_key": "train", "files": dl_manager.download_and_extract(url)},
... )
```

class datasets.Splitdatasets.Splithttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/splits.py#L406[{"name": "name", "val": ""}]
`Enum` for dataset splits.

Datasets are typically split into different subsets to be used at various stages of training and evaluation.

- `TRAIN`: the training data.
- `VALIDATION`: the validation data. If present, this is typically used as evaluation data while iterating on a model (e.g. changing hyperparameters, model architecture, etc.).
- `TEST`: the testing data. This is the data to report metrics on. Typically you do not want to use this during model iteration as you may overfit to it.
- `ALL`: the union of all defined dataset splits.

All splits, including compositions inherit from `datasets.SplitBase`.

See the guide on splits for more information.

Example:

```
>>> datasets.SplitGenerator(
...     name=datasets.Split.TRAIN,
...     gen_kwargs={"split_key": "train", "files": dl_manager.download_and extract(url)},
... ),
... datasets.SplitGenerator(
...     name=datasets.Split.VALIDATION,
...     gen_kwargs={"split_key": "validation", "files": dl_manager.download_and extract(url)},
... ),
... datasets.SplitGenerator(
...     name=datasets.Split.TEST,
...     gen_kwargs={"split_key": "test", "files": dl_manager.download_and extract(url)},
... )
```

class datasets.NamedSplitdatasets.NamedSplithttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/splits.py#L314[{"name": "name", "val": ""}]
Descriptor corresponding to a named split (train, test, ...).

Example:

Each descriptor can be composed with other using addition or slice:

```
split = datasets.Split.TRAIN.subsplit(datasets.percent[0:25]) + datasets.Split.TEST
```

The resulting split will correspond to 25% of the train split merged with
100% of the test split.

A split cannot be added twice, so the following will fail:

```
split = (
        datasets.Split.TRAIN.subsplit(datasets.percent[:25]) +
        datasets.Split.TRAIN.subsplit(datasets.percent[75:])
)  # Error
split = datasets.Split.TEST + datasets.Split.ALL  # Error
```

The slices can be applied only one time. So the following are valid:

```
split = (
        datasets.Split.TRAIN.subsplit(datasets.percent[:25]) +
        datasets.Split.TEST.subsplit(datasets.percent[:50])
)
split = (datasets.Split.TRAIN + datasets.Split.TEST).subsplit(datasets.percent[:50])
```

But this is not valid:

```
train = datasets.Split.TRAIN
test = datasets.Split.TEST
split = train.subsplit(datasets.percent[:25]).subsplit(datasets.percent[:25])
split = (train.subsplit(datasets.percent[:25]) + test).subsplit(datasets.percent[:50])
```

class datasets.NamedSplitAlldatasets.NamedSplitAllhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/splits.py#L391[]
Split corresponding to the union of all defined dataset splits.

class datasets.ReadInstructiondatasets.ReadInstructionhttps://github.com/huggingface/

datasets/blob/4.2.0/src/datasets/arrow_reader.py#L456[{"name": "split_name", "val": ""}, {"name": "rounding", "val": " = None"}, {"name": "from_", "val": " = None"}, {"name": "to", "val": " = None"}, {"name": "unit", "val": " = None"}]
Reading instruction for a dataset.

Examples:

```
# The following lines are equivalent:
ds = datasets.load_dataset('mnist', split='test[:33%]')
ds = datasets.load_dataset('mnist', split=datasets.ReadInstruction.from_spec('test[:33%]'))
ds = datasets.load_dataset('mnist', split=datasets.ReadInstruction('test', to=33, unit='%'))
ds = datasets.load_dataset('mnist', split=datasets.ReadInstruction(
'test', from_=0, to=33, unit='%'))

# The following lines are equivalent:
ds = datasets.load_dataset('mnist', split='test[:33%]+train[1:-1]')
ds = datasets.load_dataset('mnist', split=datasets.ReadInstruction.from_spec(
'test[:33%]+train[1:-1]'))
ds = datasets.load_dataset('mnist', split=(
datasets.ReadInstruction('test', to=33, unit='%') +
datasets.ReadInstruction('train', from_=1, to=-1, unit='abs')))

# The following lines are equivalent:
ds = datasets.load_dataset('mnist', split='test[:33%](pct1_dropremainder)')
ds = datasets.load_dataset('mnist', split=datasets.ReadInstruction.from_spec(
'test[:33%](pct1_dropremainder)'))
ds = datasets.load_dataset('mnist', split=datasets.ReadInstruction(
'test', from_=0, to=33, unit='%', rounding="pct1_dropremainder"))

# 10-fold validation:
tests = datasets.load_dataset(
'mnist',
[datasets.ReadInstruction('train', from_=k, to=k+10, unit='%')
for k in range(0, 100, 10)])
trains = datasets.load_dataset(
'mnist',
[datasets.ReadInstruction('train', to=k, unit='%') + datasets.ReadInstruction('train', from_=k+10,
for k in range(0, 100, 10)])
```

from_specdatasets.ReadInstruction.from_spechttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_reader.py#L536[{"name": "spec", "val": ""}]- **spec** ( `str` ) --
Split(s) + optional slice(s) to read + optional rounding
if percents are used as the slicing unit. A slice can be specified,
using absolute numbers ( `int` ) or percentages ( `int` ).0ReadInstruction instance.
Creates a `ReadInstruction` instance out of a string spec.

Examples:

```
test: test split.
test + validation: test split + validation split.
test[10:]: test split, minus its first 10 records.
test[:10%]: first 10% records of test split.
test[:20%](pct1_dropremainder): first 10% records, rounded with the pct1_dropremainder rounding.
test[:-5%]+train[40%:60%]: first 95% of test + middle 20% of train.
```

to_absolutedatasets.ReadInstruction.to_absolutehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_reader.py#L608[{"name": "name2len", "val": ""}]- **name2len** ( `dict` ) --
Associating split names to number of examples.0list of _AbsoluteInstruction instances
(corresponds to the + in spec).
Translate instruction into a list of absolute instructions.

Those absolute instructions are then to be added together.

# Versiondatasets.Version

class datasets.Versiondatasets.Versionhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/utils/version.py#L30[{"name": "version_str", "val": ": str"}, {"name": "description", "val": ": typing.Optional[str] = None"}, {"name": "major", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "minor", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "patch", "val": ": typing.Union[str, int, NoneType] = None"}]- **version_str** ( `str` ) --
The dataset version.

- **description** ( `str` ) --
  A description of what is new in this version.
- **major** ( `str` ) --

- **minor** ( `str` ) --
- **patch** ( `str` ) --0
  Dataset version `MAJOR.MINOR.PATCH`.

Example:

```
>>> VERSION = datasets.Version("1.0.0")
```