Structure your repository

To host and share your dataset, create a dataset repository on the Hugging Face Hub and upload your data files.

This guide will show you how to structure your dataset repository when you upload it.
A dataset with a supported structure and file format (`.txt` , `.csv` , `.parquet` , `.jsonl` , `.mp3` , `.jpg` , `.zip` etc.) are loaded automatically with load_dataset(), and it'll have a dataset viewer on its dataset page on the Hub.

Main use-case

The simplest dataset structure has two files: `train.csv` and `test.csv` (this works with any supported file format).

Your repository will also contain a `README.md` file, the dataset card displayed on your dataset page.

```
my_dataset_repository/
├── README.md
├── train.csv
└── test.csv
```

In this simple case, you'll get a dataset with two splits: `train` (containing examples from `train.csv`) and `test` (containing examples from `test.csv`).

Define your splits and subsets in YAML

Splits

If you have multiple files and want to define which file goes into which split, you can use the YAML `configs` field at the top of your README.md.

For example, given a repository like this one:

```
my_dataset_repository/
├── README.md
├── data.csv
└── holdout.csv
```

You can define your splits by adding the `configs` field in the YAML block at the top of your README.md:

```
---
configs:
- config_name: default
  data_files:
  - split: train
    path: "data.csv"
  - split: test
    path: "holdout.csv"
---
```

You can select multiple files per split using a list of paths:

```
my_dataset_repository/
├── README.md
├── data/
│   ├── abc.csv
│   └── def.csv
└── holdout/
    └── ghi.csv
```

```yaml
---
configs:
- config_name: default
  data_files:
  - split: train
    path:
    - "data/abc.csv"
    - "data/def.csv"
  - split: test
    path: "holdout/ghi.csv"
---
```

Or you can use glob patterns to automatically list all the files you need:

```yaml
---
configs:
- config_name: default
  data_files:
  - split: train
    path: "data/*.csv"
  - split: test
    path: "holdout/*.csv"
---
```

> [!WARNING]
> Note that `config_name` field is required even if you have a single configuration.

Configurations

Your dataset might have several subsets of data that you want to be able to load separately. In that case you can define a list of configurations inside the `configs` field in YAML:

```
my_dataset_repository/
├── README.md
├── main_data.csv
└── additional_data.csv
```

```yaml
---
configs:
- config_name: main_data
  data_files: "main_data.csv"
- config_name: additional_data
  data_files: "additional_data.csv"
---
```

Each configuration is shown separately on the Hugging Face Hub, and can be loaded by passing its name as a second parameter:

```python
from datasets import load_dataset

main_data = load_dataset("my_dataset_repository", "main_data")
additional_data = load_dataset("my_dataset_repository", "additional_data")
```

Builder parameters

Not only `data_files`, but other builder-specific parameters can be passed via YAML, allowing for more flexibility on how to load the data while not requiring any custom code. For example, define which separator to use in which configuration to load your `csv` files:

```yaml
---
configs:
- config_name: tab
  data_files: "main_data.csv"
  sep: "\t"
- config_name: comma
  data_files: "additional_data.csv"
  sep: ","
---
```

Refer to specific builders' documentation to see what configuration parameters they have.

> [!TIP]
> You can set a default configuration using `default: true`, e.g. you can run

```
main_data = load_dataset("my_dataset_repository")
```

if you set

```
- config_name: main_data
  data_files: "main_data.csv"
  default: true
```

Automatic splits detection

If no YAML is provided, 🤗 Datasets searches for certain patterns in the dataset repository to automatically infer the dataset splits.
There is an order to the patterns, beginning with the custom filename split format to treating all files as a single split if no pattern is found.

Directory name

Your data files may also be placed into different directories named `train`, `test`, and `validation` where each directory contains the data files for that split:

```
my_dataset_repository/
├── README.md
└── data/
    ├── train/
    │   └── bees.csv
    ├── test/
    │   └── more_bees.csv
    └── validation/
        └── even_more_bees.csv
```

Filename splits

If you don't have any non-traditional splits, then you can place the split name anywhere in the data file and it is automatically inferred. The only rule is that the split name must be delimited by non-word characters, like `test-file.csv` for example instead of `testfile.csv`. Supported delimiters include underscores, dashes, spaces, dots, and numbers.

For example, the following file names are all acceptable:

- train split: `train.csv`, `my_train_file.csv`, `train1.csv`
- validation split: `validation.csv`, `my_validation_file.csv`, `validation1.csv`
- test split: `test.csv`, `my_test_file.csv`, `test1.csv`

Here is an example where all the files are placed into a directory named `data`:

```
my_dataset_repository/
├── README.md
└── data/
    ├── train.csv
    ├── test.csv
    └── validation.csv
```

Custom filename split

If your dataset splits have custom names that aren't `train`, `test`, or `validation`, then you can name your data files like `data/<split_name>-xxxxx-of-xxxxx.csv`.

Here is an example with three splits, `train`, `test`, and `random`:

```
my_dataset_repository/
├── README.md
└── data/
    ├── train-00000-of-00003.csv
    ├── train-00001-of-00003.csv
    ├── train-00002-of-00003.csv
    ├── test-00000-of-00001.csv
    ├── random-00000-of-00003.csv
    ├── random-00001-of-00003.csv
    └── random-00002-of-00003.csv
```

Single split

When 🤗 Datasets can't find any of the above patterns, then it'll treat all the files as a single train split. If your dataset splits aren't loading as expected, it may be due to an incorrect pattern.

Split name keywords

There are several ways to name splits. Validation splits are sometimes called "dev", and test splits may be referred to as "eval".
These other split names are also supported, and the following keywords are equivalent:

- train, training
- validation, valid, val, dev
- test, testing, eval, evaluation

The structure below is a valid repository:

```
my_dataset_repository/
├── README.md
└── data/
    ├── training.csv
    ├── eval.csv
    └── valid.csv
```

Multiple files per split

If one of your splits comprises several files, 🤗 Datasets can still infer whether it is the train, validation, and test split from the file name.
For example, if your train and test splits span several files:

```
my_dataset_repository/
├── README.md
├── train_0.csv
├── train_1.csv
├── train_2.csv
├── train_3.csv
├── test_0.csv
└── test_1.csv
```

Make sure all the files of your `train` set have *train* in their names (same for test and validation).
Even if you add a prefix or suffix to `train` in the file name (like `my_train_file_00001.csv` for

example),
🤗 Datasets can still infer the appropriate split.

For convenience, you can also place your data files into different directories.
In this case, the split name is inferred from the directory name.

```
my_dataset_repository/
├── README.md
└── data/
    ├── train/
    │   ├── shard_0.csv
    │   ├── shard_1.csv
    │   ├── shard_2.csv
    │   └── shard_3.csv
    └── test/
        ├── shard_0.csv
        └── shard_1.csv
```