Create an audio dataset

You can share a dataset with your team or with anyone in the community by creating a dataset repository on the Hugging Face Hub:

```
from datasets import load_dataset

dataset = load_dataset("<username>/my_dataset")
```

There are several methods for creating and sharing an audio dataset:

- Create an audio dataset from local files in python with Dataset.push_to_hub(). This is an easy way that requires only a few steps in python.
- Create an audio dataset repository with the `AudioFolder` builder. This is a no-code solution for quickly creating an audio dataset with several thousand audio files.

> [!TIP]
> You can control access to your dataset by requiring users to share their contact information first. Check out the Gated datasets guide for more information about how to enable this feature on the Hub.

Local files

You can load your own dataset using the paths to your audio files. Use the cast_column() function to take a column of audio file paths, and cast it to the Audio feature:

```
>>> audio_dataset = Dataset.from_dict({"audio": ["path/to/audio_1", "path/to/audio_2", ..., "path/
>>> audio_dataset[0]["audio"]
<datasets.features._torchcodec.AudioDecoder object at 0x11642b6a0>
```

Then upload the dataset to the Hugging Face Hub using Dataset.push_to_hub():

```
audio_dataset.push_to_hub("<username>/my_dataset")
```

This will create a dataset repository containing your audio dataset:

```
my_dataset/
├── README.md
└── data/
    └── train-00000-of-00001.parquet
```

AudioFolder

The `AudioFolder` is a dataset builder designed to quickly load an audio dataset with several thousand audio files without requiring you to write any code.

> [!TIP]
> 💡 Take a look at the Split pattern hierarchy to learn more about how `AudioFolder` creates dataset splits based on your dataset repository structure.

`AudioFolder` automatically infers the class labels of your dataset based on the directory name. Store your dataset in a directory structure like:

```
folder/train/dog/golden_retriever.mp3
folder/train/dog/german_shepherd.mp3
folder/train/dog/chihuahua.mp3

folder/train/cat/maine_coon.mp3
folder/train/cat/bengal.mp3
folder/train/cat/birman.mp3
```

If the dataset follows the `AudioFolder` structure, then you can load it directly with load_dataset():

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("username/dataset_name")
```

This is equivalent to passing `audiofolder` manually in load_dataset() and the directory in `data_dir`:

```
>>> dataset = load_dataset("audiofolder", data_dir="/path/to/folder")
```

You can also use `audiofolder` to load datasets involving multiple splits. To do so, your dataset directory should have the following structure:

```
folder/train/dog/golden_retriever.mp3
folder/train/cat/maine_coon.mp3
folder/test/dog/german_shepherd.mp3
folder/test/cat/bengal.mp3
```

> [!WARNING]
> If all audio files are contained in a single directory or if they are not on the same level of directory structure, `label` column won't be added automatically. If you need it, set `drop_labels=False` explicitly.

If there is additional information you'd like to include about your dataset, like text captions or bounding boxes, add it as a `metadata.csv` file in your folder. This lets you quickly create datasets for different computer vision tasks like text captioning or object detection. You can also use a JSONL file `metadata.jsonl` or a Parquet file `metadata.parquet`.

```
folder/train/metadata.csv
folder/train/0001.mp3
folder/train/0002.mp3
folder/train/0003.mp3
```

You can also zip your audio files, and in this case each zip should contain both the audio files and the metadata

```
folder/train.zip
folder/test.zip
folder/validation.zip
```

Your `metadata.csv` file must have a `file_name` or `*_file_name` field which links audio files with their metadata:

```
file_name,additional_feature
0001.mp3,This is a first value of a text feature you added to your audio files
0002.mp3,This is a second value of a text feature you added to your audio files
0003.mp3,This is a third value of a text feature you added to your audio files
```

or using `metadata.jsonl`:

```
{"file_name": "0001.mp3", "additional_feature": "This is a first value of a text feature you added
{"file_name": "0002.mp3", "additional_feature": "This is a second value of a text feature you adde
{"file_name": "0003.mp3", "additional_feature": "This is a third value of a text feature you added
```

Here the `file_name` must be the name of the audio file next to the metadata file. More generally, it must be the relative path from the directory containing the metadata to the audio file.

It's possible to point to more than one audio in each row in your dataset, for example if both your input and output are audio files:

```
{"input_file_name": "0001.mp3", "output_file_name": "0001_output.mp3"}
{"input_file_name": "0002.mp3", "output_file_name": "0002_output.mp3"}
{"input_file_name": "0003.mp3", "output_file_name": "0003_output.mp3"}
```

You can also define lists of audio files. In that case you need to name the field `file_names` or `*_file_names`. Here is an example:

```
{"recordings_file_names": ["0001_r0.mp3", "0001_r1.mp3"], label: "same_person"}
{"recordings_file_names": ["0002_r0.mp3", "0002_r1.mp3"], label: "same_person"}
{"recordings_file_names": ["0003_r0.mp3", "0003_r1.mp3"], label: "different_person"}
```

WebDataset

The WebDataset format is based on TAR archives and is suitable for big audio datasets. Indeed you can group your audio files in TAR archives (e.g. 1GB of audio files per TAR archive) and have thousands of TAR archives:

```
folder/train/00000.tar
folder/train/00001.tar
folder/train/00002.tar
...
```

In the archives, each example is made of files sharing the same prefix:

```
e39871fd9fd74f55.mp3
e39871fd9fd74f55.json
f18b91585c4d3f3e.mp3
f18b91585c4d3f3e.json
ede6e66b2fb59aab.mp3
ede6e66b2fb59aab.json
ed600d57fcee4f94.mp3
ed600d57fcee4f94.json
...
```

You can put your audio files labels/captions/bounding boxes using JSON or text files for example.

Load your WebDataset and it will create on column per file suffix (here "mp3" and "json"):

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("webdataset", data_dir="/path/to/folder", split="train")
>>> dataset[0]["json"]
{"transcript": "Hello there !", "speaker": "Obi-Wan Kenobi"}
```

It's also possible to have several audio files per example like this:

```
e39871fd9fd74f55.input.mp3
e39871fd9fd74f55.output.mp3
e39871fd9fd74f55.json
f18b91585c4d3f3e.input.mp3
f18b91585c4d3f3e.output.mp3
f18b91585c4d3f3e.json
...
```

For more details on the WebDataset format and the python library, please check the WebDataset documentation.