Use with NumPy

This document is a quick introduction to using `datasets` with NumPy, with a particular focus on how to get
`numpy.ndarray` objects out of our datasets, and how to use them to train models based on NumPy such as `scikit-learn` models.

Dataset format

By default, datasets return regular Python objects: integers, floats, strings, lists, etc..

To get NumPy arrays instead, you can set the format of the dataset to `numpy`:

```
>>> from datasets import Dataset
>>> data = [[1, 2], [3, 4]]
>>> ds = Dataset.from_dict({"data": data})
>>> ds = ds.with_format("numpy")
>>> ds[0]
{'data': array([1, 2])}
>>> ds[:2]
{'data': array([
    [1, 2],
    [3, 4]])}
```

> [!TIP]
> A Dataset object is a wrapper of an Arrow table, which allows fast reads from arrays in the dataset to NumPy arrays.

Note that the exact same procedure applies to `DatasetDict` objects, so that when setting the format of a `DatasetDict` to `numpy`, all the `Dataset`s there will be formatted as `numpy`:

```
>>> from datasets import DatasetDict
>>> data = {"train": {"data": [[1, 2], [3, 4]]}, "test": {"data": [[5, 6], [7, 8]]}}
>>> dds = DatasetDict.from_dict(data)
>>> dds = dds.with_format("numpy")
>>> dds["train"][:2]
{'data': array([
    [1, 2],
    [3, 4]])}
```

N-dimensional arrays

If your dataset consists of N-dimensional arrays, you will see that by default they are considered as the same array if the shape is fixed:

```
>>> from datasets import Dataset
>>> data = [[[1, 2],[3, 4]], [[5, 6],[7, 8]]]  # fixed shape
>>> ds = Dataset.from_dict({"data": data})
>>> ds = ds.with_format("numpy")
>>> ds[0]
{'data': array([[1, 2],
        [3, 4]])}
```

```
>>> from datasets import Dataset
>>> data = [[[1, 2],[3]], [[4, 5, 6],[7, 8]]]  # varying shape
>>> ds = Dataset.from_dict({"data": data})
>>> ds = ds.with_format("numpy")
>>> ds[0]
{'data': array([array([1, 2]), array([3])], dtype=object)}
```

However this logic often requires slow shape comparisons and data copies.
To avoid this, you must explicitly use the `Array` feature type and specify the shape of your tensors:

```
>>> from datasets import Dataset, Features, Array2D
>>> data = [[[1, 2],[3, 4]],[[5, 6],[7, 8]]]
>>> features = Features({"data": Array2D(shape=(2, 2), dtype='int32')})
>>> ds = Dataset.from_dict({"data": data}, features=features)
>>> ds = ds.with_format("numpy")
>>> ds[0]
{'data': array([[1, 2],
        [3, 4]])}
>>> ds[:2]
{'data': array([[[1, 2],
         [3, 4]],

        [[5, 6],
         [7, 8]]])}
```

Other feature types

ClassLabel data is properly converted to arrays:

```
>>> from datasets import Dataset, Features, ClassLabel
>>> labels = [0, 0, 1]
>>> features = Features({"label": ClassLabel(names=["negative", "positive"])})
>>> ds = Dataset.from_dict({"label": labels}, features=features)
>>> ds = ds.with_format("numpy")
>>> ds[:3]
{'label': array([0, 0, 1])}
```

String and binary objects are unchanged, since NumPy only supports numbers.

The Image and Audio feature types are also supported.

> [!TIP]
> To use the Image feature type, you'll need to install the `vision` extra as `pip install datasets[vision]` .

```
>>> from datasets import Dataset, Features, Image
>>> images = ["path/to/image.png"] * 10
>>> features = Features({"image": Image()})
>>> ds = Dataset.from_dict({"image": images}, features=features)
>>> ds = ds.with_format("numpy")
>>> ds[0]["image"].shape
(512, 512, 3)
>>> ds[0]
{'image': array([[[ 255, 255, 255],
                  [ 255, 255, 255],
                  ...,
                  [ 255, 255, 255],
                  [ 255, 255, 255]]], dtype=uint8)}
>>> ds[:2]["image"].shape
(2, 512, 512, 3)
>>> ds[:2]
{'image': array([[[[ 255, 255, 255],
                  [ 255, 255, 255],
                  ...,
                  [ 255, 255, 255],
                  [ 255, 255, 255]]]], dtype=uint8)}
```

> [!TIP]
> To use the Audio feature type, you'll need to install the `audio` extra as `pip install datasets[audio]`.

```
>>> from datasets import Dataset, Features, Audio
>>> audio = ["path/to/audio.wav"] * 10
>>> features = Features({"audio": Audio()})
>>> ds = Dataset.from_dict({"audio": audio}, features=features)
>>> ds = ds.with_format("numpy")
>>> ds[0]["audio"]["array"]
array([-0.059021  , -0.03894043, -0.00735474, ...,  0.0133667 ,
                  0.01809692,  0.00268555], dtype=float32)
>>> ds[0]["audio"]["sampling_rate"]
array(44100, weak_type=True)
```

Data loading

NumPy doesn't have any built-in data loading capabilities, so you'll either need to materialize the NumPy arrays like `X, y` to use in `scikit-learn` or use a library such as PyTorch to load your data using a `DataLoader`.

Using `with_format('numpy')`

The easiest way to get NumPy arrays out of a dataset is to use the `with_format('numpy')` method. Lets assume
that we want to train a neural network on the MNIST dataset available
at the HuggingFace Hub at https://huggingface.co/datasets/mnist.

```
>>> from datasets import load_dataset
>>> ds = load_dataset("mnist")
>>> ds = ds.with_format("numpy")
>>> ds["train"][0]
{'image': array([[  0,   0,   0, ...],
                 [  0,   0,   0, ...],
                 ...,
                 [  0,   0,   0, ...],
                 [  0,   0,   0, ...]], dtype=uint8),
 'label': array(5)}
```

Once the format is set we can feed the dataset to the model based on NumPy in batches using the `Dataset.iter()`
method:

```
>>> for epoch in range(epochs):
...     for batch in ds["train"].iter(batch_size=32):
...         x, y = batch["image"], batch["label"]
...         ...
```