**parenttoc:** True

# Loading other datasets

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 10)**

Unknown directive type "currentmodule".

```
.. currentmodule:: sklearn.datasets
```

## Sample images

Scikit-learn also embeds a couple of sample JPEG images published under Creative Commons license by their authors. Those images can be useful to test algorithms and pipelines on 2D data.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 21)**

Unknown directive type "autosummary".

```
.. autosummary::

   load_sample_images
   load_sample_image
```

**Warning**

The default coding of images is based on the `uint8` dtype to spare memory. Often machine learning algorithms work best if the input is converted to a floating point representation first. Also, if you plan to use `matplotlib.pyplpt.imshow`, don't forget to scale to the range 0 - 1 as done in the following example.

**Examples:**

- :ref:`sphx_glr_auto_examples_cluster_plot_color_quantization.py`

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 42); *backlink***

Unknown interpreted text role "ref".

## Datasets in svmlight / libsvm format

scikit-learn includes utility functions for loading datasets in the svmlight / libsvm format. In this format, each line takes the form `<label> <feature-id>:<feature-value> <feature-id>:<feature-value> ...`. This format is especially suitable for sparse datasets. In this module, scipy sparse CSR matrices are used for `X` and numpy arrays are used for `y`.

You may load a dataset like as follows:

```
>>> from sklearn.datasets import load_svmlight_file
>>> X_train, y_train = load_svmlight_file("/path/to/train_dataset.txt")
...                                                       # doctest: +SKIP
```

You may also load two (or more) datasets at once:

```
>>> X_train, y_train, X_test, y_test = load_svmlight_files(
...     ("/path/to/train_dataset.txt", "/path/to/test_dataset.txt"))
...                                                       # doctest: +SKIP
```

In this case, `X_train` and `X_test` are guaranteed to have the same number of features. Another way to achieve the same result is to fix the number of features:

```
>>> X_test, y_test = load_svmlight_file(
...     "/path/to/test_dataset.txt", n_features=X_train.shape[1])
...                                                       # doctest: +SKIP
```

**Related links:**

Public datasets in svmlight / libsvm format: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

Faster API-compatible implementation: https://github.com/mblondel/svmlight-loader

### Downloading datasets from the openml.org repository

openml.org is a public repository for machine learning data and experiments, that allows everybody to upload open datasets.

The `sklearn.datasets` package is able to download datasets from the repository using the function :func:`sklearn.datasets.fetch_openml`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 95); *backlink***
>
> Unknown interpreted text role "func".

For example, to download a dataset of gene expressions in mice brains:

```
>>> from sklearn.datasets import fetch_openml
>>> mice = fetch_openml(name='miceprotein', version=4)
```

To fully specify a dataset, you need to provide a name and a version, though the version is optional, see :ref:`openml_versions` below. The dataset contains a total of 1080 examples belonging to 8 different classes:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 104); *backlink***
>
> Unknown interpreted text role "ref".

```
>>> mice.data.shape
(1080, 77)
>>> mice.target.shape
(1080,)
>>> np.unique(mice.target)
array(['c-CS-m', 'c-CS-s', 'c-SC-m', 'c-SC-s', 't-CS-m', 't-CS-s', 't-SC-m', 't-SC-s'], dtype=object)
```

You can get more information on the dataset by looking at the `DESCR` and `details` attributes:

```
>>> print(mice.DESCR) # doctest: +SKIP
**Author**: Clara Higuera, Katheleen J. Gardiner, Krzysztof J. Cios
**Source**: [UCI](https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression) - 2015
**Please cite**: Higuera C, Gardiner KJ, Cios KJ (2015) Self-Organizing
Feature Maps Identify Proteins Critical to Learning in a Mouse Model of Down
Syndrome. PLoS ONE 10(6): e0129126...

>>> mice.details # doctest: +SKIP
{'id': '40966', 'name': 'MiceProtein', 'version': '4', 'format': 'ARFF',
'upload_date': '2017-11-08T16:00:15', 'licence': 'Public',
'url': 'https://www.openml.org/data/v1/download/17928620/MiceProtein.arff',
'file_id': '17928620', 'default_target_attribute': 'class',
'row_id_attribute': 'MouseID',
'ignore_attribute': ['Genotype', 'Treatment', 'Behavior'],
'tag': ['OpenML-CC18', 'study_135', 'study_98', 'study_99'],
'visibility': 'public', 'status': 'active',
'md5_checksum': '3c479a6885bfa0438971388283a1ce32'}
```

The `DESCR` contains a free-text description of the data, while `details` contains a dictionary of meta-data stored by openml, like the dataset id. For more details, see the OpenML documentation The `data_id` of the mice protein dataset is 40966, and you can use this (or the name) to get more information on the dataset on the openml website:

```
>>> mice.url
'https://www.openml.org/d/40966'
```

The `data_id` also uniquely identifies a dataset from OpenML:

```
>>> mice = fetch_openml(data_id=40966)
>>> mice.details # doctest: +SKIP
{'id': '4550', 'name': 'MiceProtein', 'version': '1', 'format': 'ARFF',
'creator': ...,
'upload_date': '2016-02-17T14:32:49', 'licence': 'Public', 'url':
'https://www.openml.org/data/v1/download/1804243/MiceProtein.ARFF', 'file_id':
'1804243', 'default_target_attribute': 'class', 'citation': 'Higuera C,
Gardiner KJ, Cios KJ (2015) Self-Organizing Feature Maps Identify Proteins
Critical to Learning in a Mouse Model of Down Syndrome. PLoS ONE 10(6):
e0129126. [Web Link] journal.pone.0129126', 'tag': ['OpenML100', 'study_14',
```

```
'study_34'], 'visibility': 'public', 'status': 'active', 'md5_checksum':
'3c479a6885bfa0438971388283a1ce32'}
```

**Dataset Versions**

A dataset is uniquely specified by its `data_id`, but not necessarily by its name. Several different "versions" of a dataset with the same name can exist which can contain entirely different datasets. If a particular version of a dataset has been found to contain significant issues, it might be deactivated. Using a name to specify a dataset will yield the earliest version of a dataset that is still active. That means that `fetch_openml(name="miceprotein")` can yield different results at different times if earlier versions become inactive. You can see that the dataset with `data_id` 40966 that we fetched above is the first version of the "miceprotein" dataset:

```
>>> mice.details['version']  #doctest: +SKIP
'1'
```

In fact, this dataset only has one version. The iris dataset on the other hand has multiple versions:

```
>>> iris = fetch_openml(name="iris")
>>> iris.details['version']  #doctest: +SKIP
'1'
>>> iris.details['id']  #doctest: +SKIP
'61'

>>> iris_61 = fetch_openml(data_id=61)
>>> iris_61.details['version']
'1'
>>> iris_61.details['id']
'61'

>>> iris_969 = fetch_openml(data_id=969)
>>> iris_969.details['version']
'3'
>>> iris_969.details['id']
'969'
```

Specifying the dataset by the name "iris" yields the lowest version, version 1, with the `data_id` 61. To make sure you always get this exact dataset, it is safest to specify it by the dataset `data_id`. The other dataset, with `data_id` 969, is version 3 (version 2 has become inactive), and contains a binarized version of the data:

```
>>> np.unique(iris_969.target)
array(['N', 'P'], dtype=object)
```

You can also specify both the name and the version, which also uniquely identifies the dataset:

```
>>> iris_version_3 = fetch_openml(name="iris", version=3)
>>> iris_version_3.details['version']
'3'
>>> iris_version_3.details['id']
'969'
```

**References:**

- :arxiv:`Vanschoren, van Rijn, Bischl and Torgo. "OpenML: networked science in machine learning" ACM SIGKDD Explorations Newsletter, 15(2), 49-60, 2014. <1407.7722>`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst`, line 224); *backlink***
>
> Unknown interpreted text role "arxiv".

## Loading from external datasets

scikit-learn works on any numeric data stored as numpy arrays or scipy sparse matrices. Other types that are convertible to numeric arrays such as pandas DataFrame are also acceptable.

Here are some recommended ways to load standard columnar data into a format usable by scikit-learn:

- pandas.io provides tools to read data from common formats including CSV, Excel, JSON and SQL. DataFrames may also be constructed from lists of tuples or dicts. Pandas handles heterogeneous data smoothly and provides tools for manipulation and conversion into a numeric array suitable for scikit-learn.
- scipy.io specializes in binary formats often used in scientific computing context such as .mat and .arff
- numpy/routines.io for standard loading of columnar data into numpy arrays
- scikit-learn's :func:`datasets.load_svmlight_file` for the svmlight or libSVM sparse format

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-`**

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 250); *backlink*

Unknown interpreted text role "func".

- scikit-learn's :func:`datasets.load_files` for directories of text files where the name of each directory is the name of each category and each file inside of each directory corresponds to one sample from that category

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 252); *backlink*

Unknown interpreted text role "func".

For some miscellaneous data such as images, videos, and audio, you may wish to refer to:

- skimage.io or Imageio for loading images and videos into numpy arrays
- scipy.io.wavfile.read for reading WAV files into a numpy array

Categorical (or nominal) features stored as strings (common in pandas DataFrames) will need converting to numerical features using :class:`~sklearn.preprocessing.OneHotEncoder` or :class:`~sklearn.preprocessing.OrdinalEncoder` or similar. See :ref:`preprocessing`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 266); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 266); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scikit-learn-main\doc\datasets\(scikit-learn-main)(doc)(datasets)loading_other_datasets.rst, line 266); *backlink*

Unknown interpreted text role "ref".

Note: if you manage your own numerical data it is recommended to use an optimized file format such as HDF5 to reduce data load times. Various libraries such as H5Py, PyTables and pandas provides a Python interface for reading and writing data in that format.