7.2. Real world datasets

scikit-learn provides tools to load larger datasets, downloading them if necessary.

They can be loaded using the following functions:

`fetch_olivetti_faces`(*[, data_home, ...])	Load the Olivetti faces data-set from AT&T (classification).
`fetch_20newsgroups`(*[, data_home, subset, ...])	Load the filenames and data from the 20 newsgroups dataset (classification).
`fetch_20newsgroups_vectorized`(*[, subset, ...])	Load and vectorize the 20 newsgroups dataset (classification).
`fetch_lfw_people`(*[, data_home, funneled, ...])	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
`fetch_lfw_pairs`(*[, subset, data_home, ...])	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
`fetch_covtype`(*[, data_home, ...])	Load the covertype dataset (classification).
`fetch_rcv1`(*[, data_home, subset, ...])	Load the RCV1 multilabel dataset (classification).
`fetch_kddcup99`(*[, subset, data_home, ...])	Load the kddcup99 dataset (classification).
`fetch_california_housing`(*[, data_home, ...])	Load the California housing dataset (regression).

7.2.1. The Olivetti faces dataset

This dataset contains a set of face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge. The `sklearn.datasets.fetch_olivetti_faces` function is the data fetching / caching function that downloads the data archive from AT&T.

As described on the original website:

> There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

Data Set Characteristics:

Classes	40
Samples total	400
Dimensionality	4096
Features	real, between 0 and 1

The image is quantized to 256 grey levels and stored as unsigned 8-bit integers; the loader will convert these to floating point values on the interval [0, 1], which are easier to work with for many algorithms.

The “target” for this database is an integer from 0 to 39 indicating the identity of the person pictured; however, with only 10 examples per class, this relatively small dataset is more interesting from an unsupervised or semi-supervised perspective.

The original dataset consisted of 92 x 112, while the version available here consists of 64x64 images.

When using these images, please give credit to AT&T Laboratories Cambridge.

7.2.2. The 20 newsgroups text dataset

The 20 newsgroups dataset comprises around 18000 newsgroups posts on 20 topics split in two subsets: one for training (or development) and the other one for testing (or for performance evaluation). The split between the train and test set is based upon a messages posted before and after a specific date.

This module contains two loaders. The first one, `sklearn.datasets.fetch_20newsgroups`, returns a list of the raw texts that can be fed to text feature extractors such as `CountVectorizer` with custom parameters so as to extract feature vectors. The second one, `sklearn.datasets.fetch_20newsgroups_vectorized`, returns ready-to-use features, i.e., it is not necessary to use a feature extractor.

Data Set Characteristics:

Classes	20
Samples total	18846
Dimensionality	1
Features	text

7.2.2.1. Usage

The `sklearn.datasets.fetch_20newsgroups` function is a data fetching / caching functions that downloads the data archive from the original 20 newsgroups website, extracts the archive contents in the `~/scikit_learn_data/20news_home` folder and calls the `sklearn.datasets.load_files` on either the training or testing set folder, or both of them:

```
>>> from sklearn.datasets import fetch_20newsgroups
>>> newsgroups_train = fetch_20newsgroups(subset='train')

>>> from pprint import pprint
>>> pprint(list(newsgroups_train.target_names))
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.ibm.pc.hardware',
 'comp.sys.mac.hardware',
 'comp.windows.x',
 'misc.forsale',
 'rec.autos',
 'rec.motorcycles',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.med',
 'sci.space',
 'soc.religion.christian',
 'talk.politics.guns',
 'talk.politics.mideast',
 'talk.politics.misc',
 'talk.religion.misc']
```

The real data lies in the `filenames` and `target` attributes. The target attribute is the integer index of the category:

```
>>> newsgroups_train.filenames.shape
(11314,)
>>> newsgroups_train.target.shape
(11314,)
>>> newsgroups_train.target[:10]
array([ 7,  4,  4,  1, 14, 16, 13,  3,  2,  4])
```

It is possible to load only a sub-selection of the categories by passing the list of the categories to load to the `sklearn.datasets.fetch_20newsgroups` function:

```
>>> cats = ['alt.atheism', 'sci.space']
>>> newsgroups_train = fetch_20newsgroups(subset='train', categories=cats)

>>> list(newsgroups_train.target_names)
['alt.atheism', 'sci.space']
>>> newsgroups_train.filenames.shape
(1073,)
>>> newsgroups_train.target.shape
(1073,)
>>> newsgroups_train.target[:10]
array([0, 1, 1, 1, 0, 1, 1, 0, 0, 0])
```

7.2.2.2. Converting text to vectors

In order to feed predictive or clustering models with the text data, one first need to turn the text into vectors of numerical values suitable for statistical analysis. This can be achieved with the utilities of the `sklearn.feature_extraction.text` as demonstrated in the following example that extract TF-IDF vectors of unigram tokens from a subset of 20news:

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> categories = ['alt.atheism', 'talk.religion.misc',
...               'comp.graphics', 'sci.space']
>>> newsgroups_train = fetch_20newsgroups(subset='train',
...                                       categories=categories)
>>> vectorizer = TfidfVectorizer()
>>> vectors = vectorizer.fit_transform(newsgroups_train.data)
>>> vectors.shape
(2034, 34118)
```

The extracted TF-IDF vectors are very sparse, with an average of 159 non-zero components by sample in a more than 30000-dimensional space (less than .5% non-zero features):

```
>>> vectors.nnz / float(vectors.shape[0])
159.01327...
```

`sklearn.datasets.fetch_20newsgroups_vectorized` is a function which returns ready-to-use token counts features instead of file names.

7.2.2.3. Filtering text for more realistic training

It is easy for a classifier to overfit on particular things that appear in the 20 Newsgroups data, such as newsgroup headers. Many classifiers achieve very high F-scores, but their results would not generalize to other documents that aren't from this window of time.

For example, let's look at the results of a multinomial Naive Bayes classifier, which is fast to train and achieves a decent F-score:

```
>>> from sklearn.naive_bayes import MultinomialNB
>>> from sklearn import metrics
>>> newsgroups_test = fetch_20newsgroups(subset='test',
...                                      categories=categories)
>>> vectors_test = vectorizer.transform(newsgroups_test.data)
>>> clf = MultinomialNB(alpha=.01)
>>> clf.fit(vectors, newsgroups_train.target)
MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)

>>> pred = clf.predict(vectors_test)
>>> metrics.f1_score(newsgroups_test.target, pred, average='macro')
0.88213...
```

(The example Classification of text documents using sparse features shuffles the training and test data, instead of segmenting by time, and in that case multinomial Naive Bayes gets a much higher F-score of 0.88. Are you suspicious yet of what's going on inside this classifier?)

Let's take a look at what the most informative features are:

```
>>> import numpy as np
>>> def show_top10(classifier, vectorizer, categories):
...     feature_names = vectorizer.get_feature_names_out()
...     for i, category in enumerate(categories):
...         top10 = np.argsort(classifier.coef_[i])[-10:]
...         print("%s: %s" % (category, " ".join(feature_names[top10])))
...
>>> show_top10(clf, vectorizer, newsgroups_train.target_names)
alt.atheism: edu it and in you that is of to the
comp.graphics: edu in graphics it is for and of to the
sci.space: edu it that is in and space to of the
talk.religion.misc: not it you in is that and to of the
```

You can now see many things that these features have overfit to:

- Almost every group is distinguished by whether headers such as `NNTP-Posting-Host:` and `Distribution:` appear more or less often.
- Another significant feature involves whether the sender is affiliated with a university, as indicated either by their headers or their signature.
- The word "article" is a significant feature, based on how often people quote previous posts like this: "In article [article ID], [name] <[e-mail address]> wrote:"
- Other features match the names and e-mail addresses of particular people who were posting at the time.

With such an abundance of clues that distinguish newsgroups, the classifiers barely have to identify topics from text at all, and they all perform at the same high level.

For this reason, the functions that load 20 Newsgroups data provide a parameter called **remove**, telling it what kinds of information to strip out of each file. **remove** should be a tuple containing any subset of `('headers', 'footers', 'quotes')`, telling it to remove headers, signature blocks, and quotation blocks respectively.

```
>>> newsgroups_test = fetch_20newsgroups(subset='test',
...                                      remove=('headers', 'footers', 'quotes'),
...                                      categories=categories)
>>> vectors_test = vectorizer.transform(newsgroups_test.data)
>>> pred = clf.predict(vectors_test)
>>> metrics.f1_score(pred, newsgroups_test.target, average='macro')
0.77310...
```

This classifier lost over a lot of its F-score, just because we removed metadata that has little to do with topic classification. It loses even more if we also strip this metadata from the training data:

```
>>> newsgroups_train = fetch_20newsgroups(subset='train',
...                                       remove=('headers', 'footers', 'quotes'),
...                                       categories=categories)
>>> vectors = vectorizer.fit_transform(newsgroups_train.data)
>>> clf = MultinomialNB(alpha=.01)
>>> clf.fit(vectors, newsgroups_train.target)
MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)
```

```
>>> vectors_test = vectorizer.transform(newsgroups_test.data)
>>> pred = clf.predict(vectors_test)
>>> metrics.f1_score(newsgroups_test.target, pred, average='macro')
0.76995...
```

Some other classifiers cope better with this harder version of the task. Try the Sample pipeline for text feature extraction and evaluation example with and without the `remove` option to compare the results.

Data Considerations

The Cleveland Indians is a major league baseball team based in Cleveland, Ohio, USA. In December 2020, it was reported that "After several months of discussion sparked by the death of George Floyd and a national reckoning over race and colonialism, the Cleveland Indians have decided to change their name." Team owner Paul Dolan "did make it clear that the team will not make its informal nickname – the Tribe – its new team name." "It's not going to be a half-step away from the Indians," Dolan said."We will not have a Native American-themed name."

https://www.mlb.com/news/cleveland-indians-team-name-change

Recommendation

- When evaluating text classifiers on the 20 Newsgroups data, you should strip newsgroup-related metadata. In scikit-learn, you can do this by setting `remove=('headers', 'footers', 'quotes')`. The F-score will be lower because it is more realistic.
- This text dataset contains data which may be inappropriate for certain NLP applications. An example is listed in the "Data Considerations" section above. The challenge with using current text datasets in NLP for tasks such as sentence completion, clustering, and other applications is that text that is culturally biased and inflammatory will propagate biases. This should be taken into consideration when using the dataset, reviewing the output, and the bias should be documented.

Examples

- Sample pipeline for text feature extraction and evaluation
- Classification of text documents using sparse features
- FeatureHasher and DictVectorizer Comparison
- Clustering text documents using k-means

7.2.3. The Labeled Faces in the Wild face recognition dataset

This dataset is a collection of JPEG pictures of famous people collected over the internet, all details are available on the official website:

http://vis-www.cs.umass.edu/lfw/

Each picture is centered on a single face. The typical task is called Face Verification: given a pair of two pictures, a binary classifier must predict whether the two images are from the same person.

An alternative task, Face Recognition or Face Identification is: given the picture of the face of an unknown person, identify the name of the person by referring to a gallery of previously seen pictures of identified persons.

Both Face Verification and Face Recognition are tasks that are typically performed on the output of a model trained to perform Face Detection. The most popular model for Face Detection is called Viola-Jones and is implemented in the OpenCV library. The LFW faces were extracted by this face detector from various online websites.

Data Set Characteristics:

Classes	5749
Samples total	13233
Dimensionality	5828
Features	real, between 0 and 255

7.2.3.1. Usage

`scikit-learn` provides two loaders that will automatically download, cache, parse the metadata files, decode the jpeg and convert the interesting slices into memmapped numpy arrays. This dataset size is more than 200 MB. The first load typically takes more than a couple of minutes to fully decode the relevant part of the JPEG files into numpy arrays. If the dataset has been loaded once, the following times the loading times less than 200ms by using a memmapped version memoized on the disk in the `~/scikit_learn_data/lfw_home/` folder using `joblib`.

The first loader is used for the Face Identification task: a multi-class classification task (hence supervised learning):

```
>>> from sklearn.datasets import fetch_lfw_people
>>> lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)

>>> for name in lfw_people.target_names:
...     print(name)
...
Ariel Sharon
Colin Powell
Donald Rumsfeld
George W Bush
Gerhard Schroeder
Hugo Chavez
Tony Blair
```

The default slice is a rectangular shape around the face, removing most of the background:

```
>>> lfw_people.data.dtype
dtype('float32')

>>> lfw_people.data.shape
(1288, 1850)

>>> lfw_people.images.shape
(1288, 50, 37)
```

Each of the `1140` faces is assigned to a single person id in the `target` array:

```
>>> lfw_people.target.shape
(1288,)

>>> list(lfw_people.target[:10])
[5, 6, 3, 1, 0, 1, 3, 4, 3, 0]
```

The second loader is typically used for the face verification task: each sample is a pair of two picture belonging or not to the same person:

```
>>> from sklearn.datasets import fetch_lfw_pairs
>>> lfw_pairs_train = fetch_lfw_pairs(subset='train')

>>> list(lfw_pairs_train.target_names)
['Different persons', 'Same person']

>>> lfw_pairs_train.pairs.shape
(2200, 2, 62, 47)

>>> lfw_pairs_train.data.shape
(2200, 5828)

>>> lfw_pairs_train.target.shape
(2200,)
```

Both for the **`sklearn.datasets.fetch_lfw_people`** and **`sklearn.datasets.fetch_lfw_pairs`** function it is possible to get an additional dimension with the RGB color channels by passing `color=True`, in that case the shape will be `(2200, 2, 62, 47, 3)`.

The **`sklearn.datasets.fetch_lfw_pairs`** datasets is subdivided into 3 subsets: the development `train` set, the development `test` set and an evaluation `10_folds` set meant to compute performance metrics using a 10-folds cross validation scheme.

References:

Toggle Menu

- Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. University of Massachusetts, Amherst, Technical Report 07-49, October, 2007.

7.2.3.2. Examples

Faces recognition example using eigenfaces and SVMs

7.2.4. Forest covertypes

The samples in this dataset correspond to 30×30m patches of forest in the US, collected for the task of predicting each patch's cover type, i.e. the dominant species of tree. There are seven covertypes, making this a multiclass classification problem. Each sample has 54 features, described on the dataset's homepage. Some of the features are boolean indicators, while others are discrete or continuous measurements.

Data Set Characteristics:

Classes	7
Samples total	581012
Dimensionality	54
Features	int

`sklearn.datasets.fetch_covtype` will load the covertype dataset; it returns a dictionary-like 'Bunch' object with the feature matrix in the `data` member and the target values in `target`. If optional argument 'as_frame' is set to 'True', it will return `data` and `target` as pandas data frame, and there will be an additional member `frame` as well. The dataset will be downloaded from the web if necessary.

7.2.5. RCV1 dataset

Reuters Corpus Volume I (RCV1) is an archive of over 800,000 manually categorized newswire stories made available by Reuters, Ltd. for research purposes. The dataset is extensively described in [1].

Data Set Characteristics:

Classes	103
Samples total	804414
Dimensionality	47236
Features	real, between 0 and 1

`sklearn.datasets.fetch_rcv1` will load the following version: RCV1-v2, vectors, full sets, topics multilabels:

```
>>> from sklearn.datasets import fetch_rcv1
>>> rcv1 = fetch_rcv1()
```

It returns a dictionary-like object, with the following attributes:

`data`: The feature matrix is a scipy CSR sparse matrix, with 804414 samples and 47236 features. Non-zero values contains cosine-normalized, log TF-IDF vectors. A nearly chronological split is proposed in [1]: The first 23149 samples are the training set. The last 781265 samples are the testing set. This follows the official LYRL2004 chronological split. The array has 0.16% of non zero values:

```
>>> rcv1.data.shape
(804414, 47236)
```

`target`: The target values are stored in a scipy CSR sparse matrix, with 804414 samples and 103 categories. Each sample has a value of 1 in its categories, and 0 in others. The array has 3.15% of non zero values:

```
>>> rcv1.target.shape
(804414, 103)
```

`sample_id`: Each sample can be identified by its ID, ranging (with gaps) from 2286 to 810596:

```
>>> rcv1.sample_id[:3]
array([2286, 2287, 2288], dtype=uint32)
```

`target_names`: The target values are the topics of each sample. Each sample belongs to at least one topic, and to up to 17 topics. There are 103 topics, each represented by a string. Their corpus frequencies span five orders of magnitude, from 5 occurrences for 'GMIL', to 381327 for 'CCAT':

```
>>> rcv1.target_names[:3].tolist()
['E11', 'ECAT', 'M11']
```

The dataset will be downloaded from the rcv1 homepage if necessary. The compressed size is about 656 MB.

References

[1] (1,2)
Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). RCV1: A new benchmark collection for text categorization research. The Journal of Machine Learning Research, 5, 361-397.

7.2.6. Kddcup 99 dataset

The KDD Cup '99 dataset was created by processing the tcpdump portions of the 1998 DARPA Intrusion Detection System (IDS) Evaluation dataset, created by MIT Lincoln Lab [2]. The artificial data (described on the dataset's homepage) was generated using a closed network and hand-injected attacks to produce a large number of different types of attack with normal activity in the background. As the initial goal was to produce a large training set for supervised learning algorithms, there is a large proportion (80.1%) of abnormal data which is unrealistic in real world, and inappropriate for unsupervised anomaly detection which aims at detecting 'abnormal' data, i.e.:

- qualitatively different from normal data
- in large minority among the observations.

We thus transform the KDD Data set into two different data sets: SA and SF.

- SA is obtained by simply selecting all the normal data, and a small proportion of abnormal data to gives an anomaly proportion of 1%.
- SF is obtained as in [3] by simply picking up the data whose attribute logged_in is positive, thus focusing on the intrusion attack, which gives a proportion of 0.3% of attack.
- http and smtp are two subsets of SF corresponding with third feature equal to 'http' (resp. to 'smtp').

General KDD structure :

Samples total	4898431
Dimensionality	41
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

SA structure :

Samples total	976158
Dimensionality	41
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

SF structure :

Samples total	699691
Dimensionality	4
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

http structure :

Samples total	619052
Dimensionality	3
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

smtp structure :

Samples total	95373
Dimensionality	3
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

`sklearn.datasets.fetch_kddcup99` will load the kddcup99 dataset; it returns a dictionary-like object with the feature matrix in the `data` member and the target values in `target`. The "as_frame" optional argument converts `data` into a pandas DataFrame and `target` into a pandas Series. The dataset will be downloaded from the web if necessary.

[2]
Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation, Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, Kumar Das.

[3]
K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. Online unsupervised outlier detection using finite mixtures with discounting learning algorithms. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 320-324. ACM Press, 2000.

7.2.7. California Housing dataset

Data Set Characteristics:

Number of Instances:
20640
Number of Attributes:
8 numeric, predictive attributes and the target
Attribute Information:
• MedInc median income in block group • HouseAge median house age in block group • AveRooms average number of rooms per household • AveBedrms average number of bedrooms per household • Population block group population • AveOccup average number of household members • Latitude block group latitude • Longitude block group longitude
Missing Attribute Values:
None

This dataset was obtained from the StatLib repository. https://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html

The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000).

This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surprisingly large values for block groups with few households and many empty houses, such as vacation resorts.

It can be downloaded/loaded using the `sklearn.datasets.fetch_california_housing` function.

References

- Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions, Statistics and Probability Letters, 33 (1997) 291-297