9. Dataset loading utilities

The `imblearn.datasets` package is complementing the `sklearn.datasets` package. The package provides both: (i) a set of imbalanced datasets to perform systematic benchmark and (ii) a utility to create an imbalanced dataset from an original balanced dataset.

9.1. Imbalanced datasets for benchmark

`fetch_datasets` allows to fetch 27 datasets which are imbalanced and binarized. The following data sets are available:

ID	Name	Repository & Target	Ratio	#S	#F
1	ecoli	UCI, target: imU	8.6:1	336	7
2	optical_digits	UCI, target: 8	9.1:1	5,620	64
3	satimage	UCI, target: 4	9.3:1	6,435	36
4	pen_digits	UCI, target: 5	9.4:1	10,992	16
5	abalone	UCI, target: 7	9.7:1	4,177	10
6	sick_euthyroid	UCI, target: sick euthyroid	9.8:1	3,163	42
7	spectrometer	UCI, target: >=44	11:1	531	93
8	car_eval_34	UCI, target: good, v good	12:1	1,728	21
9	isolet	UCI, target: A, B	12:1	7,797	617
10	us_crime	UCI, target: >0.65	12:1	1,994	100
11	yeast_ml8	LIBSVM, target: 8	13:1	2,417	103

Edit this page

12	scene	LIBSVM, target: >one label	13:1	2,407	294
13	libras_move	UCI, target: 1	14:1	360	90
14	thyroid_sick	UCI, target: sick	15:1	3,772	52
15	coil_2000	KDD, CoIL, target: minority	16:1	9,822	85
16	arrhythmia	UCI, target: 06	17:1	452	278
17	solar_flare_m0	UCI, target: M->0	19:1	1,389	32
18	oil	UCI, target: minority	22:1	937	49
19	car_eval_4	UCI, target: vgood	26:1	1,728	21
20	wine_quality	UCI, wine, target: <=4	26:1	4,898	11
21	letter_img	UCI, target: Z	26:1	20,000	16
22	yeast_me2	UCI, target: ME2	28:1	1,484	8
23	webpage	LIBSVM, w7a, target: minority	33:1	34,780	300
24	ozone_level	UCI, ozone, data	34:1	2,536	72
25	mammography	UCI, target: minority	42:1	11,183	6
26	protein_homo	KDD CUP 2004, minority	11:1	145,751	74
27	abalone_19	UCI, target: 19	130:1	4,177	10

A specific data set can be selected as:

```
>>> from collections import Counter
>>> from imblearn.datasets import fetch_datasets
>>> ecoli = fetch_datasets()['ecoli']
>>> ecoli.data.shape
(336, 7)
>>> print(sorted(Counter(ecoli.target).items()))
[(-1, 301), (1, 35)]
```

9.2. Imbalanced generator

`make_imbalance` turns an original dataset into an imbalanced dataset. This behaviour is driven by the parameter `sampling_strategy` which behave similarly to other resampling algorithm. `sampling_strategy` can be given as a dictionary where the key corresponds to the class and the value is the number of samples in the class:

```
>>> from sklearn.datasets import load_iris
>>> from imblearn.datasets import make_imbalance
>>> iris = load_iris()
>>> sampling_strategy = {0: 20, 1: 30, 2: 40}
>>> X_imb, y_imb = make_imbalance(iris.data, iris.target,
...
sampling_strategy=sampling_strategy)
>>> sorted(Counter(y_imb).items())
[(0, 20), (1, 30), (2, 40)]
```

Note that all samples of a class are passed-through if the class is not mentioned in the dictionary:

```
>>> sampling_strategy = {0: 10}
>>> X_imb, y_imb = make_imbalance(iris.data, iris.target,
...
sampling_strategy=sampling_strategy)
>>> sorted(Counter(y_imb).items())
[(0, 10), (1, 50), (2, 50)]
```

Instead of a dictionary, a function can be defined and directly pass to `sampling_strategy`:

```
>>> def ratio_multiplier(y):
...     multiplier = {0: 0.5, 1: 0.7, 2: 0.95}
...     target_stats = Counter(y)
...     for key, value in target_stats.items():
...         target_stats[key] = int(value * multiplier[key])
...     return target_stats
>>> X_imb, y_imb = make_imbalance(iris.data, iris.target,
...
sampling_strategy=ratio_multiplier)
>>> sorted(Counter(y_imb).items())
[(0, 25), (1, 35), (2, 47)]
```

It would also work with pandas dataframe:

```
>>> from sklearn.datasets import fetch_openml
>>> df, y = fetch_openml(
...     'iris', version=1, return_X_y=True, as_frame=True)
>>> df_resampled, y_resampled = make_imbalance(
...     df, y, sampling_strategy={'Iris-setosa': 10, 'Iris-
versicolor': 20},
...     random_state=42)
>>> df_resampled.head()
        sepallength  sepalwidth  petallength  petalwidth
  13          4.3         3.0          1.1         0.1
  39          5.1         3.4          1.5         0.2
  30          4.8         3.1          1.6         0.2
  45          4.8         3.0          1.4         0.3
  17          5.1         3.5          1.4         0.3
>>> Counter(y_resampled)
Counter({'Iris-virginica': 50, 'Iris-versicolor': 20, 'Iris-
setosa': 10})
```

See Create an imbalanced dataset and How to use sampling_strategy in imbalanced-learn.

<< 8. Common pitfalls and recommended practices

10. Developer guideline >>