Version 0.8.0

1. Introduction
2. Over-sampling
3. Under-sampling
4. Combination of over- and under-sampling
5. Ensemble of samplers
6. Miscellaneous samplers
7. Metrics
8. Common pitfalls and recommended practices
9. Dataset loading utilities
10. Developer guideline
11. References

5. Ensemble of samplers

5.1. Classifier including inner balancing samplers

5.1.1. Bagging classifier

In ensemble classifiers, bagging methods build several estimators on different randomly selected subset of data. In scikit-learn, this classifier is named **`BaggingClassifier`**. However, this classifier does not allow to balance each subset of data. Therefore, when training on imbalanced data set, this classifier will favor the majority classes:

```
>>> from sklearn.datasets import make_classification
>>> X, y = make_classification(n_samples=10000,
n_features=2, n_informative=2,
...                            n_redundant=0, n_repeated=0,
n_classes=3,
...                            n_clusters_per_class=1,
...                            weights=[0.01, 0.05, 0.94],
class_sep=0.8,
...                            random_state=0)
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import balanced_accuracy_score
>>> from sklearn.ensemble import BaggingClassifier
>>> from sklearn.tree import DecisionTreeClassifier
>>> X_train, X_test, y_train, y_test = train_test_split(X,
y, random_state=0)
>>> bc =
BaggingClassifier(base_estimator=DecisionTreeClassifier(),
...                        random_state=0)
>>> bc.fit(X_train, y_train)
BaggingClassifier(...)
>>> y_pred = bc.predict(X_test)
>>> balanced_accuracy_score(y_test, y_pred)
0.77...
```

In **`BalancedBaggingClassifier`**, each bootstrap sample will be further resampled to achieve the `sampling_strategy` desired. Therefore, **`BalancedBaggingClassifier`** takes the same parameters than the scikit-learn **`BaggingClassifier`**. In addition, the sampling is controlled by the parameter `sampler` or the two parameters `sampling_strategy` and `replacement`, if one wants to use the **`RandomUnderSampler`**:

On this page
5.1. Classifier including inner balancing samplers
Edit this page

```
>>> from imblearn.ensemble import BalancedBaggingClassifier
>>> bbc =
BalancedBaggingClassifier(base_estimator=DecisionTreeClassifie

...
sampling_strategy='auto',
...                                    replacement=False,
...                                    random_state=0)
>>> bbc.fit(X_train, y_train)
BalancedBaggingClassifier(...)
>>> y_pred = bbc.predict(X_test)
>>> balanced_accuracy_score(y_test, y_pred)
0.8...
```

Changing the `sampler` will give rise to different known implementation [MO97], [HKT09], [WY09]. You can refer to the following example shows in practice these different methods: Bagging classifiers using sampler

5.1.2. Forest of randomized trees

`BalancedRandomForestClassifier` is another ensemble method in which each tree of the forest will be provided a balanced bootstrap sample [CLB+04]. This class provides all functionality of the **`RandomForestClassifier`**:

```
>>> from imblearn.ensemble import
BalancedRandomForestClassifier
>>> brf = BalancedRandomForestClassifier(n_estimators=100,
random_state=0)
>>> brf.fit(X_train, y_train)
BalancedRandomForestClassifier(...)
>>> y_pred = brf.predict(X_test)
>>> balanced_accuracy_score(y_test, y_pred)
0.8...
```

5.1.3. Boosting

Several methods taking advantage of boosting have been designed.

`RUSBoostClassifier` randomly under-sample the dataset before to perform a boosting iteration [SKVHN09]:

```
>>> from imblearn.ensemble import RUSBoostClassifier
>>> rusboost = RUSBoostClassifier(n_estimators=200,
algorithm='SAMME.R',
...                                  random_state=0)
>>> rusboost.fit(X_train, y_train)
RUSBoostClassifier(...)
>>> y_pred = rusboost.predict(X_test)
>>> balanced_accuracy_score(y_test, y_pred)
0...
```

A specific method which uses **`AdaBoostClassifier`** as learners in the bagging classifier is called “EasyEnsemble”. The **`EasyEnsembleClassifier`** allows to bag AdaBoost learners which are trained on balanced bootstrap samples [LWZ08]. Similarly to the `BalancedBaggingClassifier` API, one can construct the ensemble as:

```
>>> from imblearn.ensemble import EasyEnsembleClassifier
>>> eec = EasyEnsembleClassifier(random_state=0)
>>> eec.fit(X_train, y_train)
EasyEnsembleClassifier(...)
>>> y_pred = eec.predict(X_test)
>>> balanced_accuracy_score(y_test, y_pred)
0.6...
```

Examples

- Compare ensemble classifiers using resampling

<< 4. Combination of over- and under-sampling

6. Miscellaneous samplers >>