7. Metrics

7.1. Classification metrics

Currently, scikit-learn only offers the `sklearn.metrics.balanced_accuracy_score` (in 0.20) as metric to deal with imbalanced datasets. The module **`imblearn.metrics`** offers a couple of other metrics which are used in the literature to evaluate the quality of classifiers.

7.1.1. Sensitivity and specificity metrics

Sensitivity and specificity are metrics which are well known in medical imaging. Sensitivity (also called true positive rate or recall) is the proportion of the positive samples which is well classified while specificity (also called true negative rate) is the proportion of the negative samples which are well classified. Therefore, depending of the field of application, either the sensitivity/specificity or the precision/recall pair of metrics are used.

Currently, only the precision and recall metrics are implemented in scikit-learn. **`sensitivity_specificity_support`**, **`sensitivity_score`**, and **`specificity_score`** add the possibility to use those metrics.

7.1.2. Additional metrics specific to imbalanced datasets

The **`geometric_mean_score`** [BSanchezGR03, KM+97] is the root of the product of class-wise sensitivity. This measure tries to maximize the accuracy on each of the classes while keeping these accuracies balanced.

The **`make_index_balanced_accuracy`** [GarciaSanchezM12] can wrap any metric and give more importance to a specific class using the parameter `alpha`.

7.1.3. Macro-Averaged Mean Absolute Error (MA-MAE)

Ordinal classification is used when there is a rank among classes, for example levels of functionality or movie ratings.

Edit this page

The **macro_averaged_mean_absolute_error** [EBS09] is used for imbalanced ordinal classification. The mean absolute error is computed for each class and averaged over classes, giving an equal weight to each class.

7.1.4. Summary of important metrics

The **classification_report_imbalanced** will compute a set of metrics per class and summarize it in a table. The parameter `output_dict` allows to get a string or a Python dictionary. This dictionary can be reused to create a Pandas dataframe for instance.

7.2. Pairwise metrics

The **imblearn.metrics.pairwise** submodule implements pairwise distances that are available in scikit-learn while used in some of the methods in imbalanced-learn.

7.2.1. Value Difference Metric

The class **ValueDifferenceMetric** is implementing the Value Difference Metric proposed in [SW86]. This measure is used to compute the proximity of two samples composed of only categorical values.

Given a single feature, categories with similar correlation with the target vector will be considered closer. Let's give an example to illustrate this behaviour as given in [WM97]. `X` will be represented by a single feature which will be some color and the target will be if a sample is whether or not an apple:

```
>>> import numpy as np
>>> X = np.array(["green"] * 10 + ["red"] * 10 + ["blue"] *
10).reshape(-1, 1)
>>> y = ["apple"] * 8 + ["not apple"] * 5 + ["apple"] * 7 +
["not apple"] * 9 + ["apple"]
```

In this dataset, the categories "red" and "green" are more correlated to the target `y` and should have a smaller distance than with the category "blue". We should this behaviour. Be aware that we need to encode the `X` to work with numerical values:

```
>>> from sklearn.preprocessing import OrdinalEncoder
>>> encoder = OrdinalEncoder(dtype=np.int32)
>>> X_encoded = encoder.fit_transform(X)
```

Now, we can compute the distance between three different samples representing the different categories:

```
>>> from imblearn.metrics.pairwise import
ValueDifferenceMetric
>>> vdm = ValueDifferenceMetric().fit(X_encoded, y)
>>> X_test = np.array(["green", "red", "blue"]).reshape(-1,
1)
>>> X_test_encoded = encoder.transform(X_test)
>>> vdm.pairwise(X_test_encoded)
array([[ 0.  ,  0.04,  1.96],
       [ 0.04,  0.  ,  1.44],
       [ 1.96,  1.44,  0.  ]])
```

We see that the minimum distance happen when the categories “red” and “green” are compared. Whenever comparing with “blue”, the distance is much larger.

Mathematical formulation

The distance between feature values of two samples is defined as:

$$\delta(x, y) = \sum_{c=1}^{C} |p(c|x_f) - p(c|y_f)|^k \ ,$$

where x and y are two samples and f a given feature, C is the number of classes, $p(c|x_f)$ is the conditional probability that the output class is c given that the feature value f has the value x and k an exponent usually defined to 1 or 2.

The distance for the feature vectors X and Y is subsequently defined as:

$$\Delta(X, Y) = \sum_{f=1}^{F} \delta(X_f, Y_f)^r \ ,$$

where F is the number of feature and r an exponent usually defined equal to 1 or 2.

<< 6. Miscellaneous samplers

8. Common pitfalls and recommended practices >>