Label-based Evaluation Weighting

Estimated time: 6 minutes

One of the simplest ways to deal with a class imbalance in a dataset is to apply a weight to each class in the cost function. The minority class will be weighted more heavily than the majority class in the cost function, so that the model ends up placing equal emphasis on both classes while it is learning.

There are a few ways to accomplish this. The weightings can be calculated manually as part of the cost function itself, so that the minority class is given a higher relative weight and the majority class is given a lower relative weight.

In addition, some classification models have built-in methods for dealing with imbalanced classes.

In scikit-learn, many classification models have a class_weight parameter that can be set to achieve balanced classes. The classes are automatically weighted inversely proportional to how frequently they appear in the data set.

$W_j = n/kn_j$

where
Wj is the weight to class j, n is the number of observations, nj is the number of observations in class
j, and k is the total number of classes.

Sklearn also has a built-in utility function that will calculate weights based on class frequencies:

```
sklearn.utils.class_weight.compute_class_weight
```

There are a few parameters that can be passed to the compute class weight function. You can specify ‘balanced’, so that the function will calculate the class weights for you. You can also input a dictionary where the keys are the classes and the values are the desired class weights. If None is given, the class weights will be uniform.