How to balance a unbalanced data set?

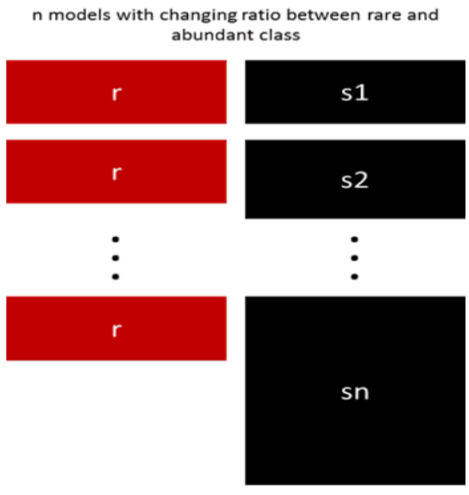
Balancing an unbalanced dataset is a common preprocessing step in machine learning, especially in scenarios where one class significantly outnumbers the other(s). Imbalanced datasets can lead to biased models that perform poorly on the minority class. Here are several techniques you can use to balance your dataset:

Resampling:

Oversampling: Increase the number of instances in the minority class by randomly duplicating existing samples or generating synthetic data points. Popular techniques include Random Oversampling, Synthetic Minority Over-sampling Technique (SMOTE), and Adaptive Synthetic Sampling (ADASYN).

Resample with Different Ratios:

The previous approach can be fine-tuned by playing with the ratio between the rare and the abundant class. The best ratio heavily depends on the data and the models that are used. But instead of training all models with the same ratio in the ensemble, it is worth trying to ensemble different ratios. So if 10 models are trained, it might make sense to have a model that has a ratio of 1:1 (rare: abundant) and another one with 1:3, or even 2:1. Depending on the model used this can influence the weight that one class gets.



Undersampling:

Reduce the number of instances in the majority class by randomly removing some samples. Be cautious not to remove too much data, which could lead to loss of information. Techniques include Random Undersampling and Tomek Links.

Generating Synthetic Data:

Techniques like SMOTE and ADASYN generate synthetic examples for the minority class by interpolating between existing samples. These methods can help balance the dataset and improve model performance.

Collecting More Data:

If possible, collecting more data for the minority class can be an effective way to balance the dataset naturally.

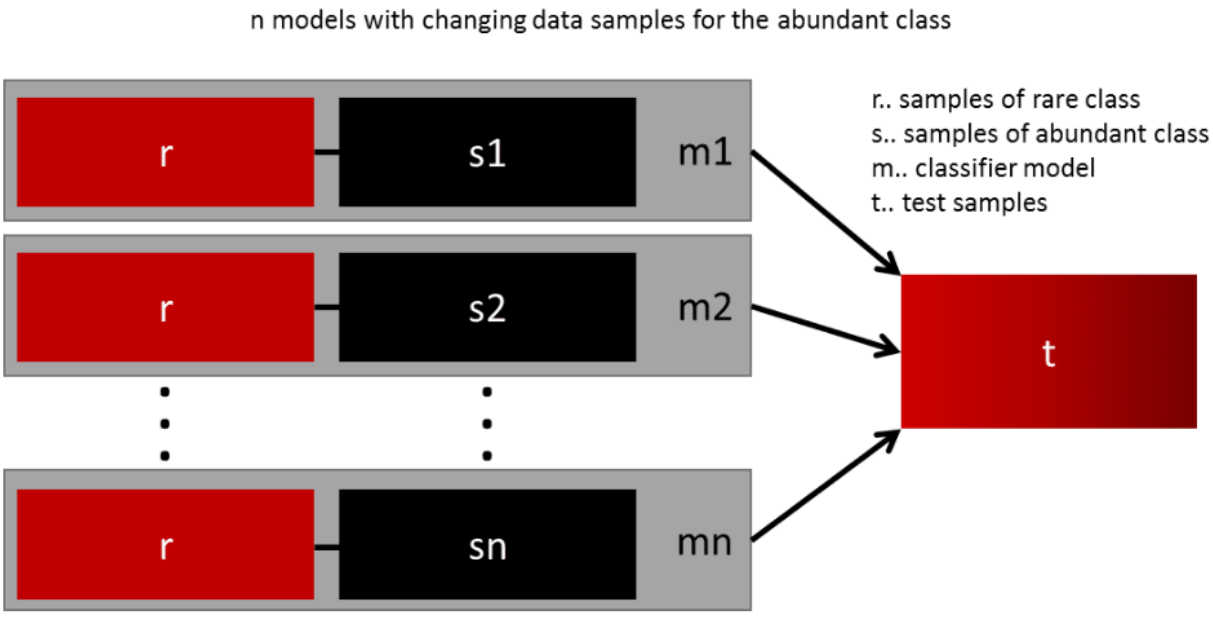
Cost-sensitive Learning:

Modify the learning algorithm to take class imbalance into account by assigning different misclassification costs to different classes. Some algorithms and libraries provide options for this, such as the class_weight parameter in scikit-learn.

Ensemble Methods: (Ensemble Different Resampled Datasets)

Use ensemble methods like Random Forests, which are less prone to class imbalance issues, as they combine predictions from multiple decision trees. The easiest way to successfully generalize a model is by using more data. The problem is that out-of-the-box classifiers like logistic regression or random forest tend to generalize by discarding the rare class. One easy best practice is building n models that use all the samples of the rare class and n-differing samples of the abundant class. Given that you want to ensemble

10 models, you would keep e.g. the 1.000 cases of the rare class and randomly sample 10.000 cases of the abundant class. Then you just split the 10.000 cases in 10 chunks and train 10 different models.



This approach is simple and perfectly horizontally scalable if you have a lot of data, since you can just train and run your models on different cluster nodes. Ensemble models also tend to generalize better, which makes this approach easy to handle.

Cluster the abundant class

Instead of relying on random samples to cover the variety of the training samples, he suggests clustering the abundant class in r groups, with r being the number of cases in r. For each group, only the medoid (centre of cluster) is kept. The model is then trained with the rare class and the medoids only.

Anomaly Detection:

Treat the minority class as an anomaly detection problem, which involves identifying rare events. Techniques such as One-Class SVM or Isolation Forests can be used in this context.

Evaluation Metrics:

When working with imbalanced datasets, focus on appropriate evaluation metrics such as precision, recall, F1-score, ROC-AUC, and precision-recall curves, rather than accuracy, as accuracy can be misleading.

Class Weighting:

Some machine learning algorithms allow you to assign different weights to different classes. By assigning higher weights to the minority class, you can give the algorithm more incentive to correctly classify those instances.

Use K-fold Cross-Validation in the Right Way / Simply do cross validation in the right way.

It is noteworthy that cross-validation should be applied properly while using over-sampling method to address imbalance problems. Keep in mind that over-sampling takes observed rare samples and applies bootstrapping to generate new random data based on a distribution function. If cross-validation is applied after over-sampling, basically what we are doing is overfitting our model to a specific artificial bootstrapping result. That is why cross-validation should always be done before over-sampling the data, just as how feature selection should be implemented. Only by resampling the data repeatedly, randomness can be introduced into the dataset to make sure that there won't be an overfitting problem.

Combining Techniques:

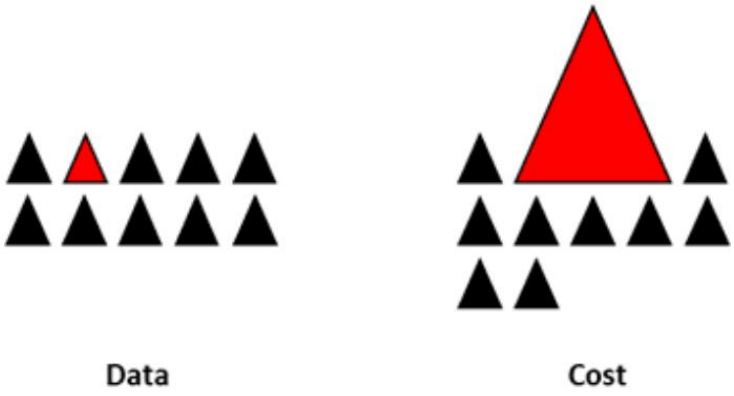
Often, a combination of oversampling, undersampling, and synthetic data generation techniques, along with careful selection of evaluation metrics and appropriate algorithms, yields the best results.

Design Your Models

All the previous methods focus on the data and keep the models as a fixed component. But in fact, there is no need to resample the data if the model is suited for imbalanced data. The famous XGBoost is already

a good starting point if the classes are not skewed too much, because it internally takes care that the bags it trains on are not imbalanced. But then again, the data is resampled, it is just happening secretly.

By designing a cost function that is penalizing wrong classification of the rare class more than wrong classifications of the abundant class, it is possible to design many models that naturally generalize in favour of the rare class. For example, tweaking an SVM to penalize wrong classifications of the rare class by the same ratio that this class is underrepresented.



Remember that the choice of balancing technique should be based on your specific dataset and problem. It's essential to assess the impact of each technique on your model's performance using appropriate cross-validation and evaluation procedures to find the most suitable approach for your task. This is not an exclusive list of techniques, but rather a starting point to handle imbalanced data. There is no best approach or model suited for all problems and it is strongly recommended to try different techniques and models to evaluate what works best. Try to be creative and combine different approaches. It is also important, to be aware that in many domains (e.g. fraud detection, real-time-bidding), where imbalanced classes occur, the “market-rules” are constantly changing. So, check if past data might have become obsolete.