**Machine Learning Tutorial**

**Universidad Nacional de Tucuman, March 2020**

**Jaume Bacardit**

**Practical 3: Preprocessing**

<https://github.com/jaumebp/ML-tutorial/tree/master/practical3>

1. **Feature selection with RFE/ReliefF**

The files practical3-rfe.py and practical3-relief.py show how to integrate feature selection within the machine learning pipeline. In this case the major code update in these files in comparison to the scripts from practical one is the use of the scikit-learn Pipeline module. This module enables the creation of chains of methods that automatically pass data from one to the other and are smart enough to appropriate handle the training and test sets. After setting a pipeline they can be trained in a single step. Each method that is added to the chain of the pipeline is identified with a string. This string is important because it becomes the prefix used in the grid search dictionary to identify the parameters that are associated to each piece of the pipeline.

Two different feature selection methods, Recursive Feature Elimination (RFE) and ReliefF are used in each of the two scripts. They are applied to the same checkerboard dataset from practical 1, and you can see how they greatly improve the performance of the RandomForest classifier used in these examples.

To use ReliefF you need to install a specific python package called skrebate (<https://github.com/EpistasisLab/scikit-rebate>).

You will see that the grid search in practice only operates to choose the number of features to select. This has been set so that the example can run quickly. Otherwise it can take, depending on the computer, more than an hour to run a broad parameter search.

You can modify this example e.g. to try other classifiers different from RandomForest, or explore some of the other feature ranking algorithms in the skrebate package.

1. **Instance selection**

The script practical3-ros.py shows how to perform some random undersampling algorithm to balance the classes of an imbalanced dataset (in which the majority class has way more examples than the minorityone). This example uses a library called imbalanced-learn. The random undersampling algorithm will take a dataset and randomly drop examples from all classes except the minority one, so that the resulting dataset has an equal number of examples as the minority class.

The dataset for this example is a very large one, so that the impact of the random undersampling is easy to observe. It is actually the protein structure prediction dataset that has been designated for the coursework of this course. This dataset was already partitioned into training and test sets (using 10-fold cross-validation) and for this example one of the pairs of training and test set is provided. These files are in a file format called ARFF, for which there is a python module (liac-arff) to open it.

In the example you will see how first the training set is reduced and then a random forest model is trained first on the original training set and then on the reduced one. Each of the two models is tested on the same test partition. Hence results between the two are comparable. You will see how in this case training from the reduced dataset is not only faster (naturally) but also leads to a model with better prediction capacity.

What can you do with this dataset?

* You can try other classifiers, to see if they can benefit equally from the random undersampling
* You can also explore some of the other rebalancing methods available in the iblearn package (<https://imbalanced-learn.readthedocs.io/en/stable/api.html>)