**Machine Learning Tutorial**

**Universidad Nacional de Tucuman, March 2020**

**Jaume Bacardit**

**Practical 1: Building machine learning pipelines**

[**https://github.com/jaumebp/ML-tutorial/tree/master/practical1**](https://github.com/jaumebp/ML-tutorial/tree/master/practical1)

This practical will focus on one particular script: practical1.py from the link above.

You will see that it is an extended version of the cross-validation/grid search script of lecture 2.

The major changes are:

1. Loading the data from a CSV file
2. In the grid search you will see an example of testing all combinations of two parameters
3. Generation of a Precision-Recall curve from the cross-validation prediction results

**The dataset**

The file checkerboard.csv contains the data that is used for this practical. The format of this file is called *comma-separated-values* (CSV):

* It is a very simple text-based format.
* The first line in the file contains the names of the attributes of the dataset separated by commas. The last attribute/column is the class label
* Each subsequent line contains one of the instances in the dataset, again with all values separated by commas
* If you want to look at the data you can open these files e.g. in Excel.

The dataset has 993 instances and 20 attributes. What is the nature of this dataset? You can guess it from the name of the file. This data contains instances that create the checkerboard pattern that was used as initial example in the first lecture of this module. That checkerboard pattern uses only two of the 20 attributes (labelled att3 and att4). What are the other 18 attributes? They are noise, irrelevant in relation to the class label, created by a gaussian function. This seems like a very simple dataset, but as you will see, it is very challenging to several ML algorithms

**Commented code**

You will see that there is quite a few lines of code that are commented (start in #). In this example by default it will use the Random Forest classifier. However, by commenting lines 24, 29, 50 and 51

classifier = RandomForestClassifier()

param\_grid = dict(max\_depth=[2,3,4],n\_estimators = [100, 200, 500])

print("Chosen max depth: {0}".format(estimator.max\_depth))

print("Chosen number of trees:{0}".format(estimator.n\_estimators))

and uncommenting lines 25, 31 and 52

classifier = KNeighborsClassifier()

param\_grid = dict(n\_neighbors=[2,3,4,5,10])

print("Number of neighbours: {0}".format(estimator.n\_neighbors))

you can switch to a K-Nearest Neighbour classifier, in which the grid search is used to identify the best number K of neighbours.

Another line of commented code is line 17:

X = data.iloc[:,[2,3]].values

If you comment line 16 and uncomment this one, then the dataset will only have two attributes, that contain the real checkerboard pattern, without any noisy irrelevant attributes.

**Plotting the PR curve for this dataset**

Lines 59-62 and 68-76 have been added to the example in order to generate the precision-recall curve that represents the performance of this classifier on the checkerboard dataset.

To generate a PR curve we need to extract, for each test instance, the probability of predicting class 1 (i.e. the second class) which is done in line 60. Then we attach these predictions across all the test sets of the cross-validation process to process them all at once. We also need to store the actual class labels for the test instances. As the cross-validation process shuffles the data we cannot rely on the original y vector.

Line 68 generates all the possible trade-offs between precision and recall based on the different thresholds that can be applied to the probabilities. These trade-offs will be the points of the curve. The rest of the code just plots the curve and saves it as a PDF file.

**What can you do with this code?**

1. Play around with the classifiers: switch from RF to KNN and see what happens in terms of performance, try to tune other parameters.
2. Add new classifiers, from the examples shown in lecture 3 or any other classifier available at scikit-learn (<https://scikit-learn.org/stable/supervised_learning.html>). Check what are their parameters and modify the grid search process accordingly
3. Switch from using the whole set of attributes to use only the two relevant ones, see which classifier was more affected by the noise of the complete dataset
4. Add other performance metrics from

<https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics>