Lab 3: ROC Analysis (`Python Version`)

Evgueni N. Smirnov
smirnov@maastrichtuniversity.nl

18 November 2021

1. Introduction

ROC analysis of probabilistic classifiers is important topic in machine learning. It is used when we need to decide whether the posterior class probabilities that the classifiers assign to test instances allow us to separate correctly the instances according to their class labels. In this lab you will implement a class `ROC` in `Python` that can be used for ROC analysis for two class problems (e.g. problems with positive and negative classes). Description of the application program interface of this class is provided in the next section.

2. Lab Tasks

We need to implement the lab tasks related to class `ROC` in the following context. A probabilistic classifer `h` has been first trained on some labeled training data and then tested on a test set of labeled test instances. For each test instance `x` classifier `h` has outputted posterior probability of the positive class and posterior probability of the negative class. The class probabilities of the test instances are saved in a `pandas.DataFrame` object `Probs` and the true class labels of the test instances are saved in a `pandas.DataFrame` object `TrueClass`. The correspondence between object `Probs` and object `TrueClass` is index-based: the positive-class probability of the `i`-th instance `x` is the `i`-th element of `P` and the true-class label of the `i`-th instance is the `i`-th element of `TrueClass`.

In the context we have just described, supply class `ROC` with the following methods:

A. Parametric constructor `ROC` that can accept two input parameters:

   a. object `Probs` that contains the estimated probabilities of the test instances for the positive class, and

   b. object `TrueClass` that contains the true class of the test instances.

   Note that `Probs` and `TrueClass` have to be saved in instance fields of `ROC` objects to allow operation of other methods in `ROC`.

B. Method `compute_ROC_coordinates` that computes the `TPr` and `FPr` coordinates of the `ROC` curve from the positive-class probabilities and true classes of the test instances (see Appendix A for pseudocode).

   One of the main problems when implementing method `compute_ROC_coordinates` is to decide how to handle test instances of opposite classes that have the same probability for the positive class. The pseudocode in Appendix A implements a strategy how to handle this problem. Propose another strategy and implement this strategy in method `compute_ROC_coordinates`.

C. Method `plot_ROC` that plots the `ROC` curve (the method assumes that we first run method `compute_ROC_coordinates`).

D. Method `compute_AUCROC` that computes the area under curve (`AUC`) of the `ROC` curve from the positive-class probabilities and true classes of the test instances (for the sake of computational efficiency, the method assumes that we first run method `compute_ROC_coordinates`).

E. Provide code to test all the methods of class `ROC` with a probabilistic classifier from `sklearn` or the `kNN` classifier from Lab 2. The data set is the `diabetes` data set.

F. (**nongraded**) Method `compute_ROC_convex_hull_coordinates` that computes the `TPr` and `FPr` coordinates of the `ROC convex hull` curve from the positive-class probabilities and true classes of the test instances.

Report: Prepare a pdf file of the Jupiter notebook with your code for class `ROC` that contains methods specified in A, B, C, D, and E. In the markdown for B describe the strategy to handle test instances of opposite classes that have the same probability for the positive class in:

(a) the pseudocode in Appendix A;

(b) your method `compute_ROC_coordinates`.

Note that implementation of the method from F is not trivial, and thus for the max grade it is enough to provide methods specified in A, B, C, D, and E only.

Appendix A: A Pseudocode of Method `compute_ROC_coordinates`

Method `compute_ROC_coordinates:`

Input: List `Probs` of class probabilities of `P` positive and `N` negative test instances,
List `TrueClass` of true class labels of `N` test instances.

Output: List `ROC_coordinates` of `(TPr, FPr)` coordinates of `ROC` curve.

```
begin
    Sort Probs in descreasing order;
    Reorder TrueClass so that the positive-class probability of instance with index i in
    Probs has the true-class label with index i in TrueClass;
    FP : = 0;
    TP := 0;
    ROC_coordinates := {};
    Previos_Prob := -∞;
    for i : = 1 to P+N do          // for each i-th instance
        begin
            if (Probs[i] ≠ Previos_Prob) then
                begin
                    Add point (FP/N, TP/P) to ROC_coordinates;
                    Previos_Prob := Probs[i]
                end
            if (TrueClass[i] is positive) then
                TP := TP + 1;
            else
                FP := FP + 1
        end;
    Add point (FP/N, TP/P) to ROC_coordinates;
    Output ROC_coordinates
end.
```

Notes:

(a) in class `ROC` the lists `Probs, TrueClass,` and `ROC_coordinates` have to be implemented as `pandas.DataFrame` objects;

(b) study the pseudcode on the small example from the lecture on model validation before actual implementation.

(c) pseudocode of standard ROC-analysis algorithms can be found at:
https://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf