SD201 TP03

1 Introduction

This lab follows on from the second lab which took place on November, 8th. You need to have completed the first and second lab before starting this one. The general instructions given in the text of the previous labs still apply to this lab. This lab is the last lab for our main project on dynamic decision trees.

1.1 Submitting and precautionary measure

The deadline to submit the whole project (all three lab sessions) is today (Wednesday) at **11.45am**. Later submissions will ***not*** be accepted. You should submit only your code. Please submit the folder of the project that contains both your code and the *results* folder but that does not contain the *input_data* folder (it is named *lab01* if you did not rename it). The project should be submitted on ecampus, under "devoir" that you can find in the section for today lab session.

You are strongly advised to make a backup copy of your project before starting this lab and throughout its completion so that, if a problem occurs that causes part of your work to fail, you can retrieve a working version.

1.2 Questions

This lab is 3 hours and 15 minutes long (from 8:30am to 11:45am). However, we will only accept new questions on the lab tasks until **10.30am**. After this time, we will answer only to the questions of students who have already been waiting, and then we will focus on troubleshooting any problems that may arise in the submission process.

1.3 Update the project for this lab

Before starting this lab, you need to replace the *main.py* file in the project on your computer with the one available on ecampus. You also need to add all the files from the *input_data* archive on ecampus to your *input_data* folder, and the files from the *expected_results* archive on ecampus to your *results/expected* folder.

2 Questions

2.1 Dynamic Tree

Question 11 (3/20) Add a method
`add_training_point(self, features: List[float], label: bool) : None`
to the `Tree` class. This method updates the tree by adding a new training point to the list of already existing training points wherever it is relevant in the tree.

Then add a method
`del_training_point(self, features: List[float], label: bool) : None`
to the `Tree` class. This method updates the tree by removing the given training point from the list of training points wherever it is relevant in the tree.

Finally, add a parameter `beta: float` with default value 0 to constructor of the `Tree` class, and update your implementation of the class so that your implementation follows the FuDyADT algorithm the pseudocode of which is given in slide 13 of the lecture on dynamic decision trees. Observe that as long as a leaf is not rebuilt, it is expected to keep the same class, even if the addition or deletion of points make the majority class change.

For testing this question, we will build a tree containing the 30% first points of each test file. Then, as long as there are following points in the file, we will call the `decide` function on the next point, then add it to the training points and delete the oldest point. The result will be the F1-score of all the predictions.