FEM31002 - Assignment 3
(Graded)

September 16, 2020

In this assignment, you are required to work on interpretation of **trained** Random Forest models with MIRCO. You will be working on German Credit Dataset. This is a binary classification problem to determine whether a person is qualified for a credit or not. Along with the current document, you will also receive a zipped file including the dataset as well as the stripped-down implementation of MIRCO. We have already discussed the main idea of MIRCO algorithm in our lectures. If you have not done so, please read the related paper for the details.

At the end the assignment, you are supposed to submit two coding files (*step1.py*, *step2.py*) and a report. Your coding files will be checked only for successful completion on CodeGrade. Note that there will be no AutoTest configuration for these code files as MIRCO requires high computational power. Your report should consist of 3 pages at most excluding the cover page.

Here are the steps to complete this assignment:

Step 1. (3 pts.) Using the methods discussed in Lecture 1, set up a numerical study to compare the performances of Random Forest (RF), MIRCO and Decision Tree (DT) algorithms on the given credit dataset. You are free to try different hyperparameter settings for RF and DT[1]. Note the following:

- Report the average performances all three methods.
- Report the average numbers of missed points by MIRCO.
- Report the average numbers of rules used all three methods and discuss whether the resulting set of rules with MIRCO or DT is more interpretable.
- Submit your Python code in a file, called *step1.py*, to CodeGrade.

Step 2. (7 pts.) You will notice that Chvatal's greedy set covering heuristic (Algorithm 1 in the paper) used in MIRCO takes a long time to complete. Any ideas on how to improve the time without compromising too much from accuracy? Implement your idea and compare the average performance of MIRCO with your algorithm against the results that you obtained in Step 1. Submit your Python code in a file, called *step2.py*, to CodeGrade.

[1]Note that MIRCO can take a long time to complete especially when `max_depth` parameter of Random Forest algorithm is large.

Step 3. (6 pts.) In its current form, MIRCO itself is not a classification algorithm (classifier). As described in the paper, the rules extracted by MIRCO from a trained Random Forest do not necessarily cover all the feature space. Can you propose an algorithm that makes MIRCO a classifier? Write down the pseudo code of your algorithm.

Step 4. (4 pts.) Write an additional function to simplify the resulting set of rules obtained with MIRCO. One simplification could be compacting the clauses forming a rule. For instance, MIRCO may return:

```
RULE 21:
==> x[0] <= 2.50
==> x[1] > 16.50
==> x[1] <= 43.50
==> x[1] <= 39.00
==> x[6] > 1.50
```

Clearly, this can be simplified to

```
RULE 21:
==> x[0] <= 2.50
==> 16.50 < x[1] <= 39.00
==> x[6] > 1.50
```

Can you think of any further simplifications?