The final project has to be worked on **by yourself**. The purpose of this project is for you to you get hands-on experience on most topics of the course and to show that you can present and explain the results of your work.
You will have to write a paper of **max. 4 pages** with an optional appendix for longer tables and figures with additional information. In the material you can find a template for the paper. The paper needs to be submitted before March 31st 2023 by mail to:
`janek.thomas@stat.uni-muenchen`.
Your grade is determined by the paper (40%), quality of code (10%) and the oral exam (50%).

Automated Machine Learning for Imbalanced Binary Classification

Imbalanced classification is an important special setting of regular classification. One class (typically the most important class) occurs much less frequent than other classes A brief introduction to imbalanced classification can be found here:
`https://link.springer.com/article/10.1007/s13748-016-0094-0`.
Your task is to build an AutoML tool for imbalanced binary classification outperforming a simple random forest baseline on a subset of a standard benchmark:

For simplicity we defined a subset that is already uploaded to OpenML and can easily be downloaded:

Name	data_id	% small class
JapaneseVowels	976	0.16
optdigits	980	0.10
ipums_la_98-small	1002	0.10
ipums_la_00-small	1018	0.06
pendigits	1019	0.10
page-blocks	1021	0.10
sylva_prior	1040	0.6
jm1	1053	0.19
musk	1116	0.15
rl	41160	0.16

You are allowed a **maximum runtime of 1 hour** per dataset. In the end, you should convince us that you outperformed a random forest baseline by your approach. To this end, you could consider the following:

- Data-level methods that modify the collection of examples to balance distributions and/or remove difficult samples (c.f. Krawczyk, B. (2016)).
- Algorithm-level methods that directly modify existing learning algorithms to alleviate the bias towards majority objects and adapt them to mining data with skewed distributions (c.f. Krawczyk, B. (2016)).
- Hybrid methods that combine the advantages of two previous groups (c.f. Krawczyk, B. (2016)).
- Data preprocessing and feature engineering.
- Ensembling and stacking.
- Efficient optimization, either by Bayesian optimization or evolutionary algorithms.
- Multi-fidelity optimization.

You are not allowing to (re-)use existing AutoML frameworks. To evaluate your approach you have to choose the way you evaluate as well; you could consider the following:

- Measure and compare against the performance of the random forest;
- Use a reasonable number of folds for crossvalidation, both internally for the AutoML system and externally for evaluation;
- Plot the performance of your AutoML approach over time;

You are allowed to use all scripts and tools you already know from the exercises; however, you are not limited to them. Overall, you should respect the following constraints:

- **Metric:**
  - The final performance has to be measured in terms of **balanced accuracy**.
- **Experimental Constraints:**
  - Your code for making design decisions should run no longer than 1 hour per data set on a single machine.
  - You can use any kind of hardware that is available to you. For example, you could also consider using Google Colab (which repeatedly offers a VM with a GPU for at most 12h for free) or Amazon SageMaker (which offers quite some resources for free if you are a first-time customer). *Don't forget to state in your paper what kind of hardware you used!*

General constraints for code submissions Please adhere to these rules to make our and your life easier! We will deduct points if your solution does not fulfill the following:

- We will use exclusively Python 3.7+ or R 4.0+.
- We expect scripts that conduct the experiments and creates results and visualizations.
- Add comments and docstrings, so we can understand your solution.
- The `README` describes how to install requirements or provides addition information.
- Add required additional packages to `requirements.txt` (for Python) or `DESCRIPTION` (for R). Explain in your `README` what this package does, why you use that package and provide a link to it's documentation or GitHub page.

Grading:

- Paper: (at most 80):
  - Abstract summmarizing the main facts of the paper: 5 points
  - Convincing motivation of the main idea in the introduction: 10 points
  - Sound and complete explanation of the approach: 20 points
  - Solution idea in general: 10 points for approaches from the lecture, further 10 points for ideas beyond the lecture
  - Thorough, insightful, and reproducible experiments: 20 points
  - Language quality (typos, gramma): 5 points
- Code (at most 20):
  - Well documented: 4 points
  - DocStrings: 4 points
  - Code quality: 4 points
  - Requirements: 4 points
  - Reproducibility[1]: 4 points
- Oral exam (at most 100):
  - Correct answers to question regarding lecture content

[1] `https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf`