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%).

Meta-Learning for XGBoost hyperparameters

Extreme Gradient Boosting (XGBoost) is an efficient implementation of gradient boosted trees. It is highly configurable by a number of hyperparameters that influence the performance. In this project we provide you with a large dataset of XGBoost configurations evaluated on 99 classification data sets.
Your task is to develop a meta-learning approach using that data to learn which configurations to use on a set of 20 new classification problems.
Your meta-learner has to outperform the following default configuration:

Hyperparameter	Value
`num_rounds`	464
`eta`	0.0082
`subsample`	0.982
`max_depth`	11
`min_child_weight`	3.30
`colsample_bytree`	0.975
`colsample_bylevel`	0.900
`lambda`	0.06068
`alpha`	0.00235
`gamma`	0

For on overview of what these hyperparameters do see:
`https://xgboost.readthedocs.io/en/stable/parameter.html`.

NOTE: the names of these hyperparameters might not match the names of the hyperparameters of the XGBoost API you are using, e.g., `num_rounds` refers to `n_estimators` in the python (scikit-learn) XGBoost API whereas it refers to `nrounds` in the R (mlr3) XGBoost API. Be careful and check the documentation!

These 20 OpenML tasks should be used for evaluation of your solution:
16, 22, 31, 2074, 2079, 3493, 3907, 3913, 9950, 9952, 9971, 10106, 14954, 14970, 146212, 146825, 167119, 167125, 168332, 168336

NOTE: These tasks naturally come with a 10-fold CV resampling. To speed things up, you can and should train only on the train set of the first fold and evaluate on the test set of the first fold!

The meta data can be found here:
`https://syncandshare.lrz.de/getlink/fiV9MfvupyNzWpT99M5RhFh2/`

- `xgboost_meta_data.csv` contains 3.386.866 evaluations of configurations across 99 tasks.
- `features.csv` contains simple pre-computed meta features for the evaluated datasets.

To this end, you could consider the following:

- Try out different performance predictors (EPMs).
- Evaluate which meta-features are useful.
- Evaluate which hyperparameters have the most influence on the performance.
- Try to find the simplest meta-learning approach that beats the default configuration.
- Check if you can find an improved static default configuration.

Important: Do not overfit on the 20 test tasks. E.g., to evaluate the performance of different EPMs use the precomputed data.

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 performance has to be measured in terms of Area Under the Receiver Operating Characteristic Curve (ROC AUC). For multi-class classification use weighted average one vs. rest (OVR) aggregation.
- **Experimental Constraints:**
  - You are only allowed to evaluate a single configuration per test task.
  - 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`