Exam Overview

The exam involves finding the best tuned and trained model or pipeline for a given data modality, based on its performance on a specific test set.

Each group (2-3 members) must choose *one* modality to demonstrate their AutoML solution on. The designed solution *must* incorporate components from the lecture weeks and/or provided bonus material, and relevant literature. Teams have complete freedom in using code from the exercises from this course and available open-sourced code.

The provided template code serves only as a *starter kit* and does *not* need to be used in the final submission. The test scores provided for various datasets are only meant as a *reference* or *signal* for your initial attempt. The exam evaluation will prioritize methodological rigor and scientific soundness over achieving the highest test scores.

`Exam Summary`:

- *Phase I, Dev Stage* [July 1-August 6]: Given multiple development datasets, develop an AutoML system such that, given a new dataset, a one-click pipeline execution returns predictions on the test split.
- *Phase II, Test Stage* [July 25 - August 6]: Given the complete train split, and only the test features (without labels) of the test split, of a new dataset, generate test predictions and submit via Github to obtain the test score.
- *Submission & presentation* [August 6, 12/13]: Submit code and poster by `August 6, 23:59 CET`; present poster on `August 12/13`.

The following sections detail requirements for each modality, followed by grading guidelines and helpful tips.

Good luck!

For quick navigation: Tabular | Vision | Text | Grading | Submission | Tips | Contact

---

Modality I: Tabular data

Given: The (input, target) of the training and test splits of 5 regression datasets, meant to be used for developing your solution.
`GitHub Code Template`: Contains code to access data and a reference structure, only as an example.
For final code and poster submissions, Github Classroom links will be made available later along with the final test data.
`Poster template`.

Goal: Find the best obtainable R^2 score[1] on the *final-exam-dataset*, where the (input, target) of training and (input, —) of the test split will be released.

Scope: You have complete freedom to choose and design the components of your AutoML solution. The primary goal is to find a tuned model that gives strong test predictions on a new dataset without any changes to the method itself. Given tabular data modality, the design space could be represented as one or more of (and not limited to this list):

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#r2-score`

- Choice of algorithm(s)
  - Could be part of the search space or a fixed design
  - Clear motivation and explanation are required for the choices made here
  - Can include traditional ML (tree-based, linear) and/or deep learning approaches (including tabular foundation models)
- Choice of hyperparameter space
  - Could be part of the search space or fixed design
  - Clear motivation and explanation required for the choices made here, including the bounds and possible defaults
- Data preprocessing and cleaning
  - Could be part of the search space or be fixed as part of the pipeline
  - Handling missing values, outliers, categorical encoding, scaling
- Feature engineering and selection
  - Automated feature creation, transformation, and selection strategies
  - Each choice brings its own nuance and should be considered and explained
- Choice and design of fidelity dimension(s)
  - Any variable that can define a suitable proxy (dataset size, number of features, training epochs, etc.)
  - Each such choice brings its own nuance and should be considered and explained
- External source(s)
  - Can be any well-motivated, publicly available, or any other setup that can be trained offline
  - Such examples could be meta-learned configurations, transfer learning, etc.

Summary: Construct an AutoML pipeline of your choice that, given a training split of a dataset, can yield a model that, when evaluated on a hidden test set, gives strong performance. The designed solution should:

- Be a (*almost*) one-click solution that yields a tuned, trained model for inference on a test set.
- Optionally offers a window of user interaction: at the beginning (e.g., expert prior), once in the middle (staging, active learning, etc.), or at the end (e.g., selection from Pareto front), or LLM API interaction. Multiple modes of interaction can be combined with suitable justification on what is being *automated*.
- The total cost of obtaining a trained model, given a new dataset, should not exceed 24 `hours` (including intervention costs, if applicable).
- Any meta learned component or pretraining that does not use any of the data provided (in Phase I) can be excluded from the above budget.

Reference baseline(s): All performances reported on the 5 Phase I datasets and the final exam dataset will be scored from a weak, off-the-shelf AutoML solution available, for an undisclosed budget and compute invested. A final performance worse than this number can be compensated for by a reasonably innovative and scientific approach.

Given the strength of existing tabular methods, you are free to construct auxiliary objectives (such as model size or inference cost) or pick a strong baseline and demonstrate how you could improve upon it!

Modality II: Image data

Given: The (input, target) of the training and test splits of 3 classification datasets, meant to be used for developing your solution.
`GitHub Code Template`: Contains code to access data and a reference structure, only as an example.
For final code and poster submissions, Github Classroom links will be made available later along with the final test data.
`Poster template`.

Goal: Find the best obtainable *top-1 accuracy* on the *final-exam-dataset*, where the (input, target) of training and (input, —) of the test split will be released.

Scope: Complete freedom to choose and design the components required to achieve the primary goal of the best hyperparameter setting to train the final model. Given image data, the design space could be represented as:

- Choice of architecture(s)
  - Could be part of search space or fixed design
  - Clear motivation and explanation required for the choices made here
  - Can use publicly available pretrained vision models for finetuning
- Choice of hyperparameter space
  - Could be part of search space or fixed design
  - Clear motivation and explanation required for the choices made here, including the bounds and possible defaults
- (Optional) Space of possible augmentations
  - Could be part of search space or be fixed as part of the pipeline
- (Optional) Choice and design of fidelity dimension(s)
  - Any variable that can define a suitable proxy or approximation of the final target problem (epochs, dataset, image resolution, model size, etc.)
  - Each such choice brings its own nuance and should be considered and explained
- (Optional) External source(s)
  - Can be any well-motivated, publicly available, or any other setup that can be trained offline with data not part of this exam
  - Such examples could be, and not limited to, expert prior inputs, meta-trained surrogates, etc.

Summary: Construct an AutoML pipeline of your choice, that given a training split of a dataset, can yield a model that when evaluated on a hidden test set, gives strong performance. The designed solution should:

- Be a (*almost*) one-click solution that yields a tuned, trained model for inference on a test set.
- Optionally offers a window of user interaction: at the beginning (e.g., expert prior), once in the middle (staging, active learning, etc.), or at the end (e.g., selection from Pareto front), or LLM API interaction. Multiple modes of interaction can be combined with suitable justification on what is being *automated*.
- The total cost of obtaining test predictions, given a new dataset, should not exceed 24 `hours` (including intervention costs, if applicable).

- Any meta learned component or pretraining that does not use any of the data provided (in Phase I) can be excluded from the above budget.
- The time or cost for the `Final model training` and `Final model evaluation` is **excluded** from the above budget.

Reference baseline(s): All performances reported on the 3 Phase I datasets and the final exam dataset will be scored from a weak, off-the-shelf AutoML solution available, for an undisclosed budget and compute invested. A final performance worse than this number can be compensated for by a reasonably innovative and scientific approach.

Given the vast spectrum of methods and compute requirements in vision, you have additional flexibility to explore auxiliary objectives (such as model size or inference cost), or improve multiple metrics (precision, F1, etc.), or leverage the 24-hour constraint and limited GPU(s) as fidelity dimensions.

Modality III: Textual data

Given: The (input, target) of the training and test splits of 4 NLP classification datasets, meant to be used for developing your solution. `GitHub Code Template`: Contains code to access data and a reference structure, only as an example.
For final code and poster submissions, Github Classroom links will be made available later along with the final test data.
`Poster template`.

Goal: Find the best obtainable *top-1 accuracy* on the *final-exam-dataset*, where the (input, target) of training and (input, —) of the test split will be released.

Scope: Complete freedom to choose and design the components required to achieve the primary goal of obtaining a tuned, trained final model. Given text data, the design space could be represented as:

- Choice of text representation and preprocessing
  - Could be part of search space or fixed design
  - Traditional approaches: TF-IDF vectorization, n-gram features, bag-of-words
  - Modern approaches: pretrained embeddings, contextualized representations
  - Text preprocessing: tokenization strategies, normalization, handling of special tokens
- Choice of architecture(s)
  - Could be part of search space or fixed design
  - Clear motivation and explanation required for the choices made here
  - Traditional ML: MLP on featurized inputs, SVM, tree-based methods
  - Deep learning: LSTM, GRU, CNN, Transformer-based architectures
  - Can use publicly available pretrained language models for finetuning
- Choice of hyperparameter space
  - Could be part of search space or fixed design
  - Clear motivation and explanation required for the choices made here, including the bounds and possible defaults
  - For finetuning approaches: fraction of layers to tune, learning rates, dropout rates

  - For traditional approaches: regularization parameters, feature selection thresholds
- Text augmentation strategies
  - Could be part of the search space or be fixed as part of the pipeline
  - Synonym replacement, back-translation, paraphrasing, noise injection
- Choice and design of fidelity dimension(s)
  - Any variable that can define a suitable proxy or approximation of the final target problem (sequence length, vocabulary size, training epochs, dataset size, model depth, etc.)
  - Each such choice brings its own nuance and should be considered and explained
- External source(s)
  - Can be any well-motivated, publicly available, or any other setup that can be trained offline with data not part of this exam
  - Such examples could be pretrained language models, external text corpora for transfer learning, meta-learned configurations, etc.

Summary: Construct an AutoML pipeline of your choice, that given a training split of a dataset, can yield a model that when evaluated on a hidden test set, gives strong performance. The designed solution should:

- Be a (*almost*) one-click solution that yields a tuned, trained model for inference on a test set.
- Optionally offers a window of user interaction: at the beginning (e.g., expert prior), once in the middle (staging, active learning, etc.), or at the end (e.g., selection from Pareto front), or LLM API interaction. Multiple modes of interaction can be combined with suitable justification on what is being *automated*.
- The total cost of obtaining test predictions, given a new dataset, should not exceed 24 `hours` (including intervention costs, if applicable).
- Any meta learned component or pretraining that does not use any of the data provided (in Phase I) can be excluded from the above budget.
- The time or cost for the `Final model training` and `Final model evaluation` is **excluded** from the above budget.

Reference baseline(s): All performances reported on the 4 Phase I datasets and the final exam dataset will be scored from a weak, off-the-shelf AutoML solution available, for an undisclosed budget and compute invested. A final performance worse than this number can be compensated for by a reasonably innovative and scientific approach.

Given the diverse landscape of text processing methods—from traditional feature engineering to modern transformer architectures—you have flexibility to explore hybrid approaches, leverage the computational constraints as design considerations, or focus on specific aspects like efficiency, interpretability, or robustness across different text domains.

Grading guidelines

The grade primarily depends on the poster presentation by the group members. Therefore, it also depends on the poster content and thereby the approach taken. Below, we discuss briefly what would be our evaluation emphasis to the approach.
The evaluation prioritizes **methodological innovation and scientific rigor** over raw test performance. While surpassing the reference baseline is encouraged, the primary assessment focuses:

Scientific rigor in HPO methodology (`High Weightage`):

- **Experimental design and validation**: Clear hypotheses, appropriate baselines, proper train/validation/test splits, and statistical significance testing where applicable
- **Search strategy justification**: Well-motivated choices for hyperparameter spaces, search algorithms, and stopping criteria with theoretical or empirical backing

Creative use of AutoML concepts (`High Weightage`):

- **Novel combinations**: Innovative integration of concepts from lectures (multi-fidelity optimization, meta-learning, neural architecture search, algorithm selection, etc.)
- **Problem-specific adaptations**: Thoughtful modifications of existing methods to address unique challenges of the given modality, and the dataset meta properties
- **Multi-objective considerations**: Creative framing beyond accuracy, such as efficiency, interpretability, robustness, etc.
- **Advanced techniques**: Implementation of sophisticated AutoML concepts (e.g., learning curves extrapolation, transfer learning, ensemble methods)
- **Auxiliary innovations**: Novel fidelity dimensions, search space designs, or evaluation metrics

Appropriate use of compute resources:

- **Efficiency documentation**: Clear tracking of computational costs (wall-clock time, GPU hours, memory) with strategic allocation of the 24-hour budget across pipeline components
- **Cost-performance analysis**: Analysis of trade-offs between computational budget and model performance, including scalability considerations and resource-aware design choices

Technical submission requirements:

- Complete, runnable code with clear structure and documentation
- Comprehensive list of dependencies with specific version numbers
- Single command to run the complete AutoML pipeline on a new dataset, returning either a trained model or a configuration to train a model
- (If the latter, above) Separate command to retrain and evaluate a specific configuration
- Configuration files or scripts that enable the reproduction of reported results

⚠ **Important**: A solution with modest test performance but exceptional methodology, creativity, and resource efficiency will score higher than a high-performing solution with poor scientific practices or trivial technical contributions.

⚠ **NOTE**: A large count of submissions for test score evaluation will be considered as tuning over the test set, which is a forbidden practice, affecting grades *negatively*.

⚠ **NOTE**: Each group must ensure it is clear who contributed to which parts of the work, as individual grades may vary accordingly.

For poster:

- Present with good practices for empirical science followed
- Denote on poster the weeks from which concepts were used for the designed solution
- Report compute or resources used overall

Submission Guidelines

- **Code submission**: Submit complete, runnable code via Github Classroom by 06.08.2025 (23:59). Follow the repository README.md for exact requirements. Code must execute with a single command given proper dependencies.
- **Test predictions**: Generate and submit test predictions in the format specified in the Github Classroom README.md. Include the resulting test score in your final poster.
- **Poster submission**: Upload your final poster by 06.08.2025 (23:59). Include method description, analysis of the Phase I datasets, test score on *final-exam-dataset*, and documentation of computational resources and experimental protocols.
- **Poster session requirements**: Bring a printed poster to the exam session. A missing poster during the exam is equivalent to a fail. Incomplete code submission will negatively impact your grade.

| Poster Template |

Recommendation and Tips

Some thoughts that could assist in choosing your data modality:

- **Tabular data**: Plethora of available open-source solutions; active research area for specialized, constrained data; multiple sub-components to innovate on; much lower compute requirements; faster development cycles; relative improvements can be diminishing.
- **Image data**: Diverse directions available; many families of AutoML methods applicable; plenty of scope for novelty; high compute requirements; many possibilities for auxiliary objectives; rich sources of fidelities; can leverage pre-training.
- **Text data**: Active area of research[2]; data pre-processing, encoding, embedding crucial; could have high compute requirements; necessitate using lower fidelity; suitable scope for LLM application.

NOTE: Available compute and time should be heavily-weighted factors in the above decision.

Technical tips:

- **Start small, test robustly, then scale**: Use Phase I datasets to build a benchmarking setup for method development, or meta-learned approaches
- **Establish baselines**: Compare against literature and simple methods to gauge improvement potential
- **Reproducible measurements**: Base decisions on adequate repetitions and variance estimates
- **Diverse initial approaches**: Start with simple but varied methods; analyze what works based on dataset properties

Consider using HPOSuite-like benchmarking libraries for faster development cycles and baseline comparisons, depending on your modality and hypotheses.

Common pitfalls to avoid:

- **Overfitting to practice data**: Don't overoptimize for the specific characteristics of the Phase I practice datasets. Design for generalization.
- **Ignoring baseline comparisons**: Always compare against simple baselines (random search, default hyperparameters) to demonstrate the value of your AutoML approach.
- **Poor time management**: Utilize all team members' time well, keeping adequate time to generate test scores, and make the poster; *always* remember Murphy's law[3].
- **Incomplete documentation**: Track everything from the beginning. Retroactively documenting compute costs and design choices is error-prone and time-consuming. Moreover, well-documented and commented code is essential for bug-free collaborative coding.
- **Test score rabbit hole**: Remember that this is an exam and thus its scope is not in prioritising *only* for a state-of-the-art score.

Remember: The goal is to build a *general-purpose AutoML system* that works well on unseen data, not to overfit to the practice datasets. Focus on principled design choices that you can justify scientifically.

[2]For feasibility purposes, the datasets in this modality are from classical NLP, solvable without large-scale transformers.
[3]`https://en.wikipedia.org/wiki/Murphy%27s_law`

Compute suggestions

There is no restriction on the nature of compute used. The budget allowed for the project is compute independent. Only the final dataset run should respect the `24-hour` limit.

Moreover, teams working under compute constraints are encouraged to develop innovative solutions that demonstrate how AutoML can still outperform manual tuning even with limited resources!

Some free options we recommend for compute:

- `Google Colaboratory`
- State provided clusters for all students:
  - `NEMO2`: can request up to 2k CPUs
  - `Helix`: can request CPUs and GPUs

  This might require a few days to get access, and we recommend that at least one member from each team explore the protocol for access as soon as possible (but prevent this from being a time sink).
- Check the `TF cluster pool` for possible access to a GPU and increase of allowed disk storage

Any team using private compute resources that include their own hardware or paid cloud services are completely free to do so. However, every resource used MUST be reported in the final presentation.

⚠ **WARNING**: Please DO NOT wait till the last week to find or resolve compute resources!

Contact and TA support

The usual exercise sessions on Thursdays, post-lecture time and Discord will continue to be possible mediums for students to ask their questions.
To ask questions about different modalities, please use the Discord classroom-forum with the specific tags listed below. This helps ensure your questions are visible and useful to other students as well.

Topic	**Tag**
Modality I: Tabular data	Exam: Tabular
Modality II: Image data	Exam: Image
Modality III: Textual data	Exam: Text
GitHub related issues	Exam: Github issues

For organizational and administrative queries, please email: `automl-lecture-orga25@cs.uni-freiburg.de`.

This assignment is due on 06.08.2025 (23:59). Submit your solution by pushing your code and the poster to your group's repository. Teams of at most 3 students are allowed. No single-member teams are allowed, except under special circumstances handled individually.