# AutoML - introduction

Katarzyna Woźnica

25th October 2021

# Our goal

## Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning

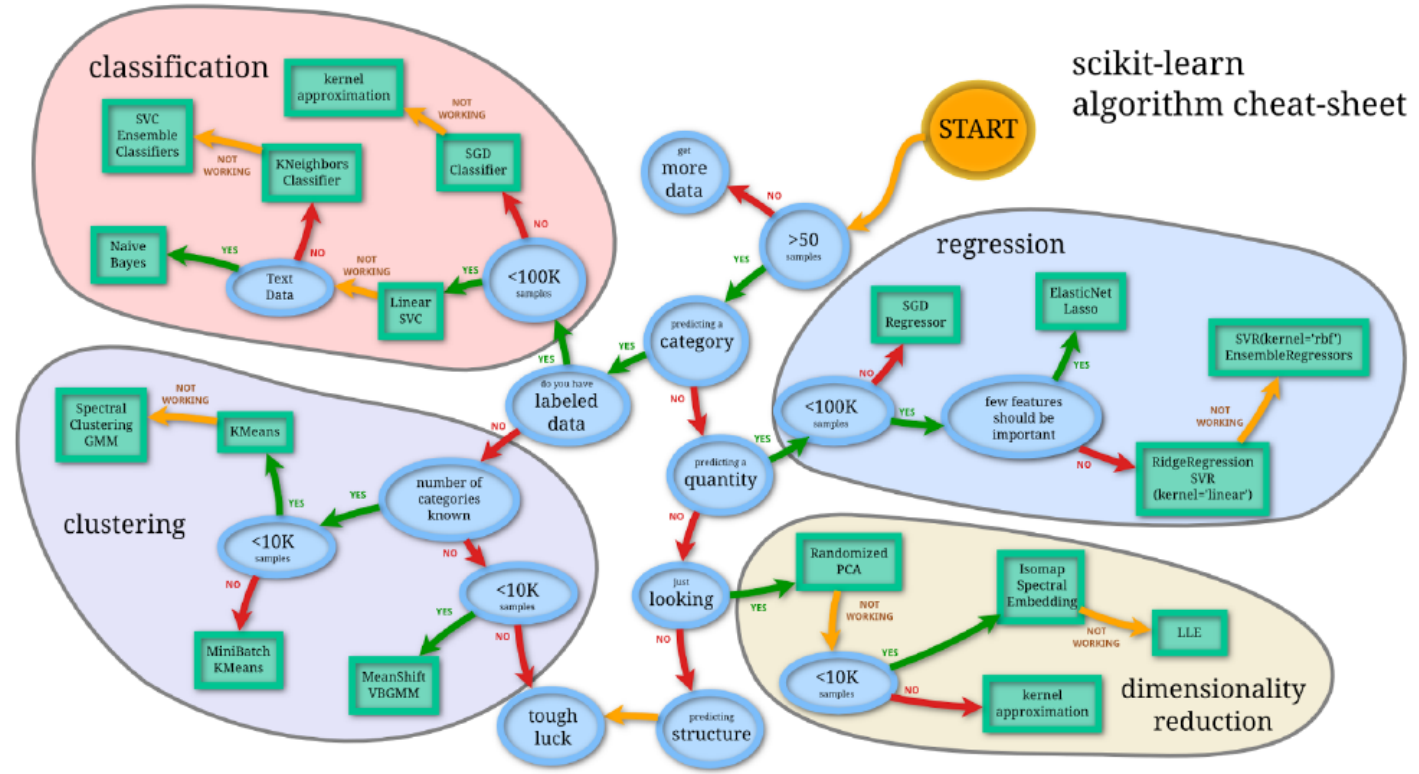
**Matthias Feurer**[1] FEURERM@CS.UNI-FREIBURG.DE
**Katharina Eggensperger**[1] EGGENSPK@CS.UNI-FREIBURG.DE
**Stefan Falkner**[2] STEFAN.FALKNER@DE.BOSCH.COM
**Marius Lindauer**[3] LINDAUER@TNT.UNI-HANNOVER.DE
**Frank Hutter**[1,2] FH@CS.UNI-FREIBURG.DE

[1] *Department of Computer Science, Albert-Ludwigs-Universität Freiburg*
[2] *Bosch Center for Artificial Intelligence, Renningen, Germany*
[3] *Institute of Information Processing, Leibniz University Hannover*

# Plan for today

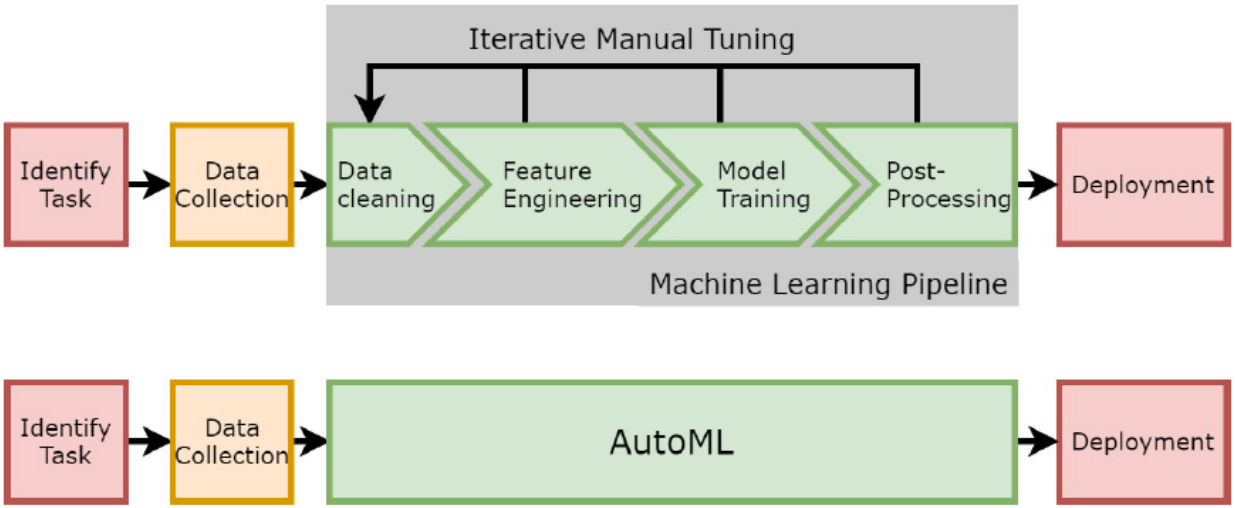
1. Motivation - get excited about AutoML
2. Theoretical problem statement
3. ChaLearn challenge
4. Auto-sklearn 1.0 and what we can expect in Auto-sklearn 2.0

# Why we need AutoML

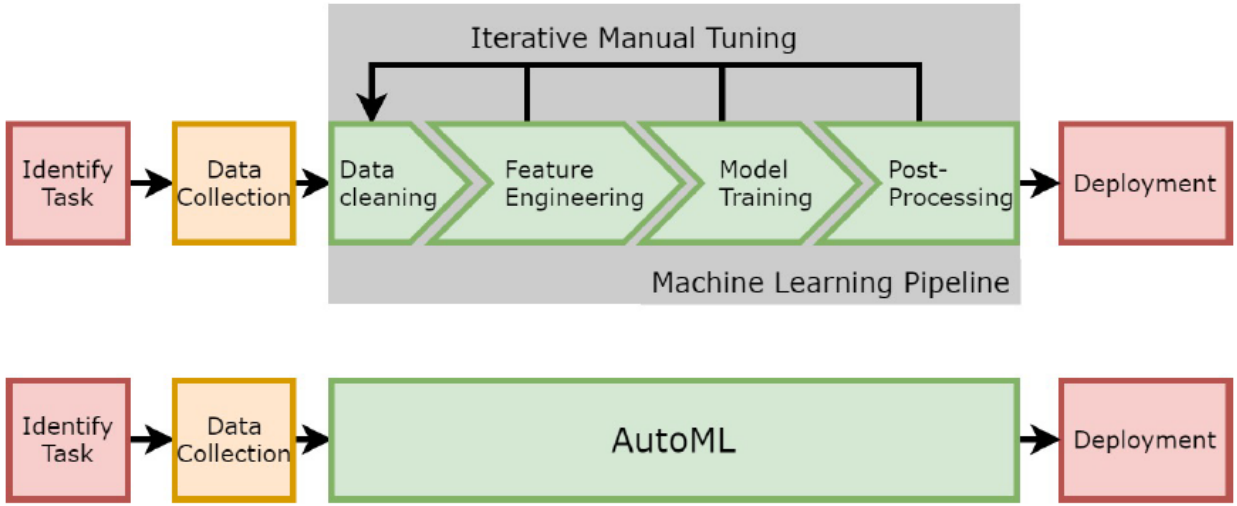



Basics in machine learning are not hard to grasp but Achieving state-of-the-art performance is quite hard.

# AutoML


Iterative Manual Tuning
Identify Task
Data Collection
Data cleaning
Feature Engineering
Model Training
Post-Processing
Deployment
Machine Learning Pipeline
Identify Task
Data Collection
AutoML
Deployment

# AutoML




With AutoML, we

- support ML users
- improve the efficiency of developing new ML applications
- reduce the required ML-expertise
- might achieve better performance than developers w/o AutoML

AutoML enables:

- more efficient research
  - AutoML has shown on subproblems to outperform human experts
- more systematic research
  - humans tend to be unsystematic which leads to errors
- more reproducible research
  - human's unsystematic approaches cannot be reproduced, but AutoML is systematic
- broader use of ML also in other disciplines
  - ML should not be limited to computer scientists;
  - the most amazing applications of ML are often done by either interdisciplinary teams or even non-computer scientists

# Problem statement

$\mathcal{P}(\mathcal{D})$ - distribution of datasets
$P_d = P_d(\mathbf{x}, y) \sim \mathcal{P}(\mathcal{D})$ - dataset distribution
$\mathcal{D}_d \sim P_d$ - sample from dataset

$\mathcal{M}_\lambda$ - pipeline (algorithm) hyperparametrized by $\lambda \in \Lambda$
$\mathcal{L}(.,.)$ - loss or cost function (single or multiple objective)

# AutoML - definition

In AutoML we want to generate pipeline $\mathcal{M}_\lambda : \mathbf{x} \rightarrow y$ automatically produces predictions minimizing the expected generalization error $GE$ for data from distribution $\mathcal{P}_d$:

$$GE(\mathcal{M}_\lambda) = \mathbf{E}_{(\mathbf{x},y)\sim P_d}\left[\mathcal{L}(\mathcal{M}_\lambda(\mathbf{x}), y)\right]$$

Approx. on $\mathcal{D}_d = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$: $\hat{GE}(\mathcal{M}_\lambda, \mathcal{D}_d) = \frac{1}{n}\sum_{i=1}^{n}\left[\mathcal{L}(\mathcal{M}_\lambda(\mathbf{x}_i), y_i)\right]$

Thus,

$$\mathcal{M}_{\lambda^*} \in argmin_{\lambda\in\Lambda}\hat{GE}(\mathcal{M}_\lambda, \mathcal{D}_{train}),$$

where $\hat{GE}(\mathcal{M}_\lambda, \mathcal{D}_{train})$ is estimated by K-fold crossvalidation.

# Types of AutoML

1. HPO - Hyperparameter Optimization
2. CASH - Combined Algorithm Selection and Hyperparameter optimization/ Full Model Selection
3. NAS - Neural Architecture Search

# Generalization of AutoML

An AutoML system $\mathcal{A} : \mathcal{D} \rightarrow \mathcal{M}_{\lambda}^{\mathcal{D}}$ should not only perform well on a single dataset but on the entire distribution over datasets.

$$GE(\mathcal{A}) = \mathbf{E}_{\mathcal{D}_d \sim \mathcal{P}(\mathcal{D})} \left[ \hat{GE} \left( \mathcal{A}(\mathcal{D}_d), \mathcal{D}_d \right) \right]$$

Time-bounded AutoML

$$\mathcal{M}_{\lambda^*} \in argmin_{\lambda \in \Lambda} \hat{GE}(\mathcal{M}_{\lambda}, \mathcal{D}_{train}),$$

where $\sum t_{\lambda_i} < T$

# Methods - HPO

- Grid search
  - curse of dimensionality
  - easily parallelizable
  - demands of expertise in HPO
- Random search
  - computationally expensive
  - useful baseline
  - easily parallelizable
- Bayesian Optimization (SMBO,SMAC)
  - state-of-the-art (uninformed approaches)
  - does not scale well with parallel resources

# Meta-learning

- We can often extract additional characteristics for each task, called meta-features

  landmarkers - computed by running several fast ML algorithms on the dataset (this can capture different properties of the dataset)
- Meta-features can be used to define a similarity measure between two tasks
- Based on similarity, we can transfer information from the most similar tasks to new task

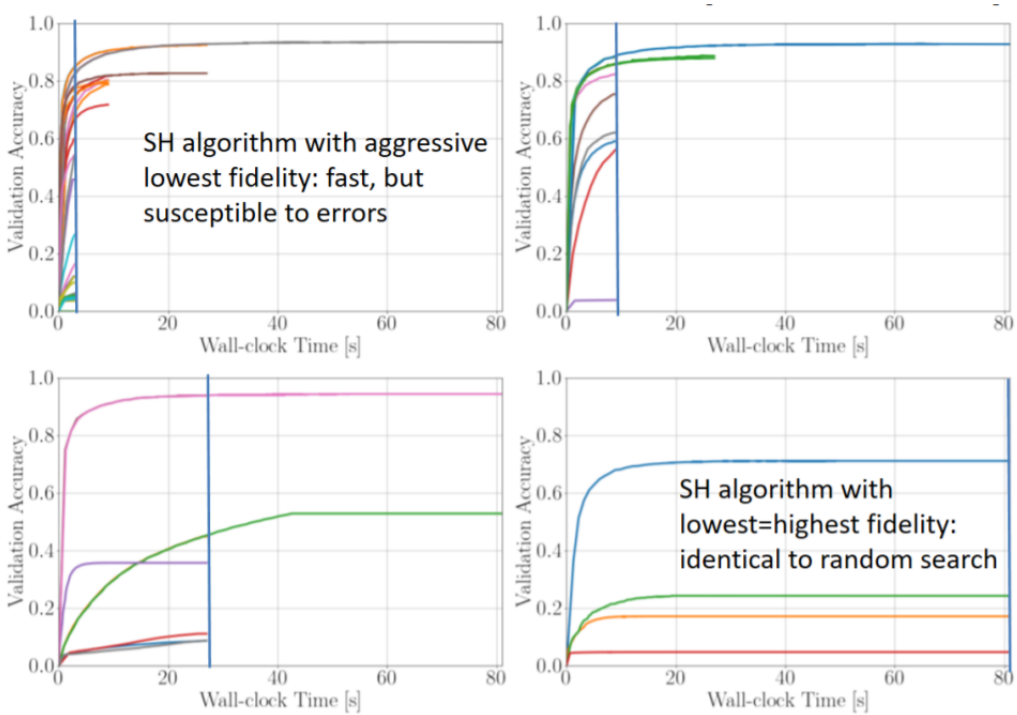
Warmstarting or task-independent recommendations

# Multi-fidelity Optimization

Exploit cheap approximations of an expensive blackbox function $\rightarrow$ afford more configurations

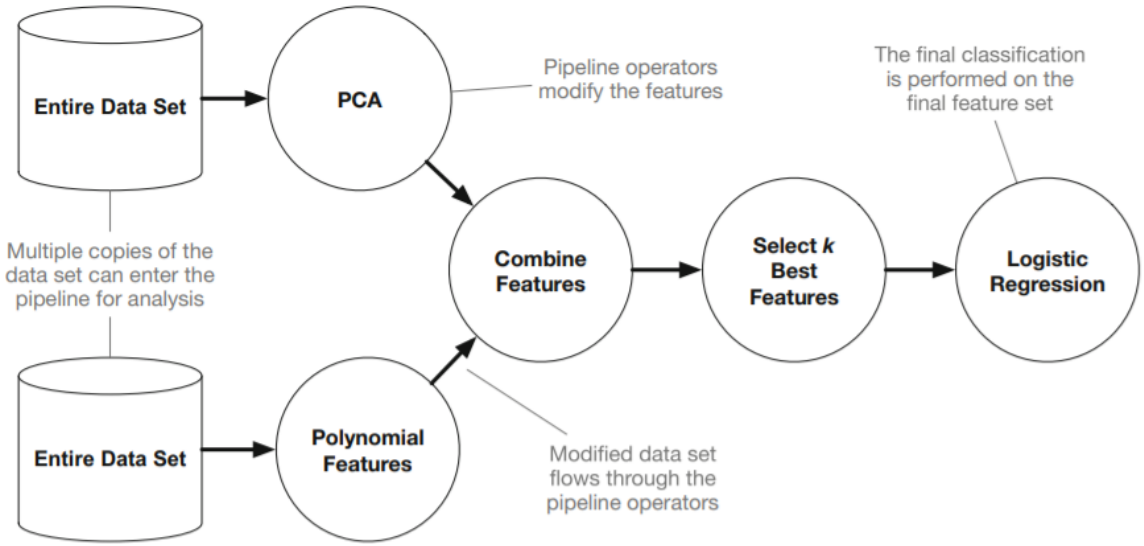
Idea: eliminate poor configurations early, allocate more resources to promising ones.

- Successive Halving
- Hyperband (+BO = BOHB)



# AutoML systems

1. AutoWEKA (2013) - SMAC + WEKA
2. Hyperopt (2014) - scikit-learn
3. Auto-sklearn 1.0 (2015) - extension of Auto-WEKA, scikit-learn
4. TPOT: Tree-Based Pipeline Optimization Tool (2016)- scikit-learn
5. Auto-sklearn 2.0 (2021) - scikit-learn



# Challenge Motivation

**The ambition of the ChaLearn AutoML challenge series is to channel the energy of the ML community to reduce step by step the need for human intervention in applying ML to a wide variety of practical problems.**

To assess the current state of AutoML and, more importantly, to foster progress in AutoML, ChaLearn conducted a series of AutoML challenges, which evaluated AutoML systems in a systematic way under rigid time and memory constraints.

# ChaLearn

ChaLearn is non-profit organization

1. AutoML 2015-2016 – winner Auto-sklearn six phases (30 datasets)
2. AutoML 2018 (4 months) – winner Auto-sklearn 2.0
   two phases: development phase (5 datasets) and blind test (5 datasets)
3. AutoDL 2019
4. MetaDL - now

Terminology
task = dataset + metric

# Constraining problem

1. **Supervised** binary classification learning problems.
2. **Feature vector representations.**
3. Homogeneous datasets (same distribution in the training, validation, and test set).
4. Medium size datasets of less than 200 MBytes.
5. Limited computer resources with execution times of less than **20 min per dataset** on an 6 compute workers. Each compute worker was equipped with 2 cores x86_64 and 8 GB of memory.

# Challenges (1)

- Different data distributions: the intrinsic/geometrical complexity of the dataset.
- Different tasks: regression, binary classification, multi-class classification, multi-label classification.
- Different scoring metrics: AUC, BAC, MSE, F1, etc.
- Class balance: Balanced or unbalanced class proportions.
- Sparsity: Full matrices or sparse matrices.

# Challenges (2)

- Missing values: Presence or absence of missing values.
- Categorical variables: Presence or absence of categorical variables.
- Irrelevant variables: Presence or absence of additional irrelevant variables (distractors).
- Number $P_{tr}$ of training examples: Small or large number of training examples.
- Number $N$ of variables/features: Small or large number of variables.
- RatioPtrNof the training data matrix:$P_{tr} \geq N$, $P_{tr} = N$ or $P_{tr} \leq N$.

# Datasets

**Table 10.3 Datasets of the 2018 AutoML challenge.** All tasks are binary classification problems. The metric is the AUC for all tasks. The time budget is also the same for all datasets (1200 s). Phase 1 was the development phase and phase 2 the final "blind test" phase

| Phase | DATASET | Cbal | Sparse | Miss | Cat | Irr | Pte | Pva | Ptr | N | Ptr/N |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 ADA | 1 | 0.67 | 0 | 0 | 0 | 41,471 | 415 | 4147 | 48 | 86.39 |
| 1 | 2 ARCENE | 0.22 | 0.54 | 0 | 0 | 0 | 700 | 100 | 100 | 10,000 | 0.01 |
| 1 | 3 GINA | 1 | 0.03 | 0.31 | 0 | 0 | 31,532 | 315 | 3153 | 970 | 3.25 |
| 1 | 4 GUILLERMO | 0.33 | 0.53 | 0 | 0 | 0 | 5000 | 5000 | 20,000 | 4296 | 4.65 |
| 1 | 5 RL | 0.10 | 0 | 0.11 | 1 | 0 | 24,803 | 0 | 31,406 | 22 | 1427.5 |
| 2 | 1 PM | 0.01 | 0 | 0.11 | 1 | 0 | 20,000 | 0 | 29,964 | 89 | 224.71 |
| 2 | 2 RH | 0.04 | 0.41 | 0 | 1 | 0 | 28,544 | 0 | 31,498 | 76 | 414.44 |
| 2 | 3 RI | 0.02 | 0.09 | 0.26 | 1 | 0 | 26,744 | 0 | 30,562 | 113 | 270.46 |
| 2 | 4 RICCARDO | 0.67 | 0.51 | 0 | 0 | 0 | 5000 | 5000 | 20,000 | 4296 | 4.65 |
| 2 | 5 RM | 0.001 | 0 | 0.11 | 1 | 0 | 26,961 | 0 | 28,278 | 89 | 317.73 |

# Ensembling

- Method of dealing with over-fitting and any-time learning
- All trained predictors are usually incorporated in the ensemble. For instance, if cross-validation is used, the predictors of all folds are directly incorporated in the ensemble
- The approaches differ in the way they weigh the contributions of the various predictors. Some methods use the same weight for all predictors, some methods assess the weights of the predictors as part of learning.

# Model space: Homogeneous vs. heterogeneous

- Most participants opted for searching a relatively large model space, including a wide variety of models found in the scikit-learn library.
- Yet, one of the strongest entrants (the Intel team) submitted results simply obtained with a boosted decision tree. It suffices to use just one machine learning approach that is a universal approximator to be able to learn anything, given enough training data.

# Preprocessing and feature selection

the top-ranking participants did not focus on preprocessing. They relied on the simple heuristics provided in the starting kit: replacing missing values by the median and adding a missingness indicator variable, one-hot-encoding of categorical variables.

# Auto-sklearn 1.0

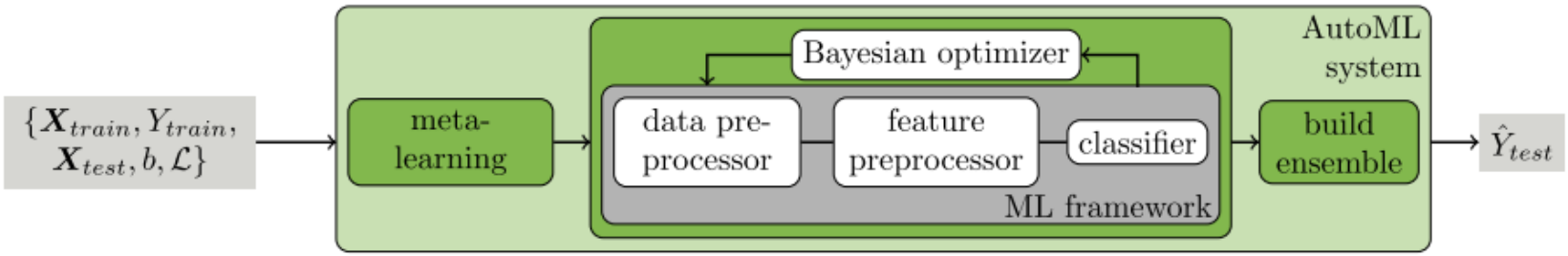



Figure 1: Our improved AutoML approach. We add two components to Bayesian hyperparameter optimization of an ML framework: meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations evaluated during optimization.

Improvements:

- warmstart BO with meta-learning
- automatic ensemble

# Meta-learning

- In an offline phase, for each machine learning dataset in a dataset repository, we evaluated a set of meta-features and used Bayesian optimization to determine and store an instantiation of the given ML framework with strong empirical performance for that dataset.
- Then, given a new dataset, we compute its meta-features, rank all datasets by their L1 distance to D in meta-feature space and select the stored ML framework instantiations for remaining third).

# Automatic Ensemble

- A uniformly weighted ensemble of the models found by Bayesian optimization does not work well
- Ensemble selection: a greedy procedure that starts from an empty ensemble and then iteratively adds the model that maximizes ensemble validation performance (with uniform weight, but allowing for repetitions)

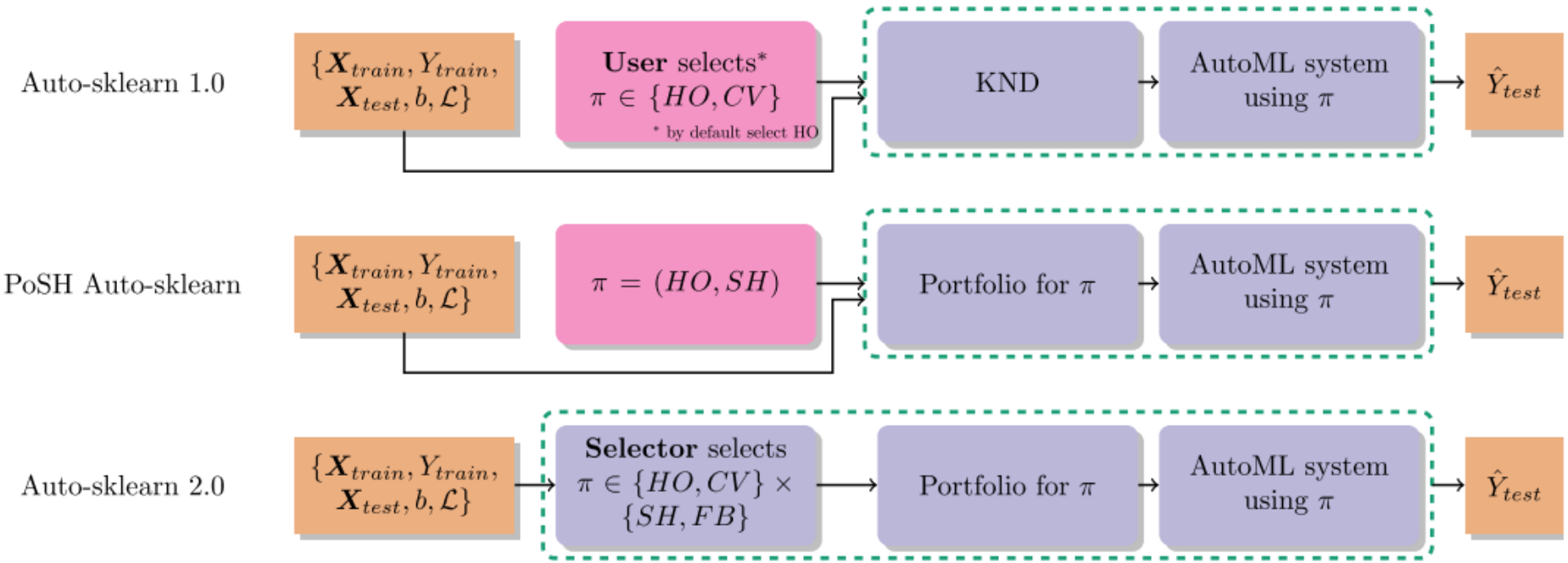
# Auto-sklearn 2.0

$$GE(\mathcal{A}) = \mathbf{E}_{\mathcal{D}_d \sim \mathcal{P}(\mathcal{D})} \left[ \hat{GE} \left( \mathcal{A}(\mathcal{D}_d), \mathcal{D}_d \right) \right]$$

Instead of using a single, fixed AutoML system $\mathcal{A}$, we will introduce optimization policies $\pi$, used in a run, which can be used to configure an AutoML system for specific use cases.
Further plan:

- How to construct $\pi$ manually?
- a novel system for designing $\pi$ from data and then extend this to a (learned) mapping

# Auto-sklearn 2.0



# Sources


- Guyon, I., Sun-Hosoya, L., Boullé, M., Escalante, H. J., Escalera, S., Liu, Z., ... & Viegas, E. (2019). Analysis of the AutoML Challenge Series. Automated Machine Learning, 177.
- Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). Automated machine learning: methods, systems, challenges (p. 219). Springer Nature.
- Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning.
- Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. Efficient and Robust Automated Machine Learning, 2015.