General Introduction to AutoML

Marius Lindauer

Leibniz Universität Hannover
Germany

/ LindauerMarius
m.lindauer@ai.uni-hannover.de

Katharina Eggensperger

Eberhard Karls Universität Tübingen
Germany

/ KEggensperger
katharina.eggensperger@uni-tuebingen.de

Questions?

Let’s use sli.do

https://app.sli.do/event/fptZhnBcuqozfF7NQ3gBm2

Story Line Today

- The Big Picture
- The Team
- … and what you will learn this week

- The Challenges
- The Big Picture II
- The Risks

Note: This lecture is based on the free online lecture "Automated Machine Learning" at https://learn.ki-campus.org/courses/automl-luh2021
- Big Picture
- Evaluation of ML Models

The Big Picture

>> What is this about?

Machine learning is this …

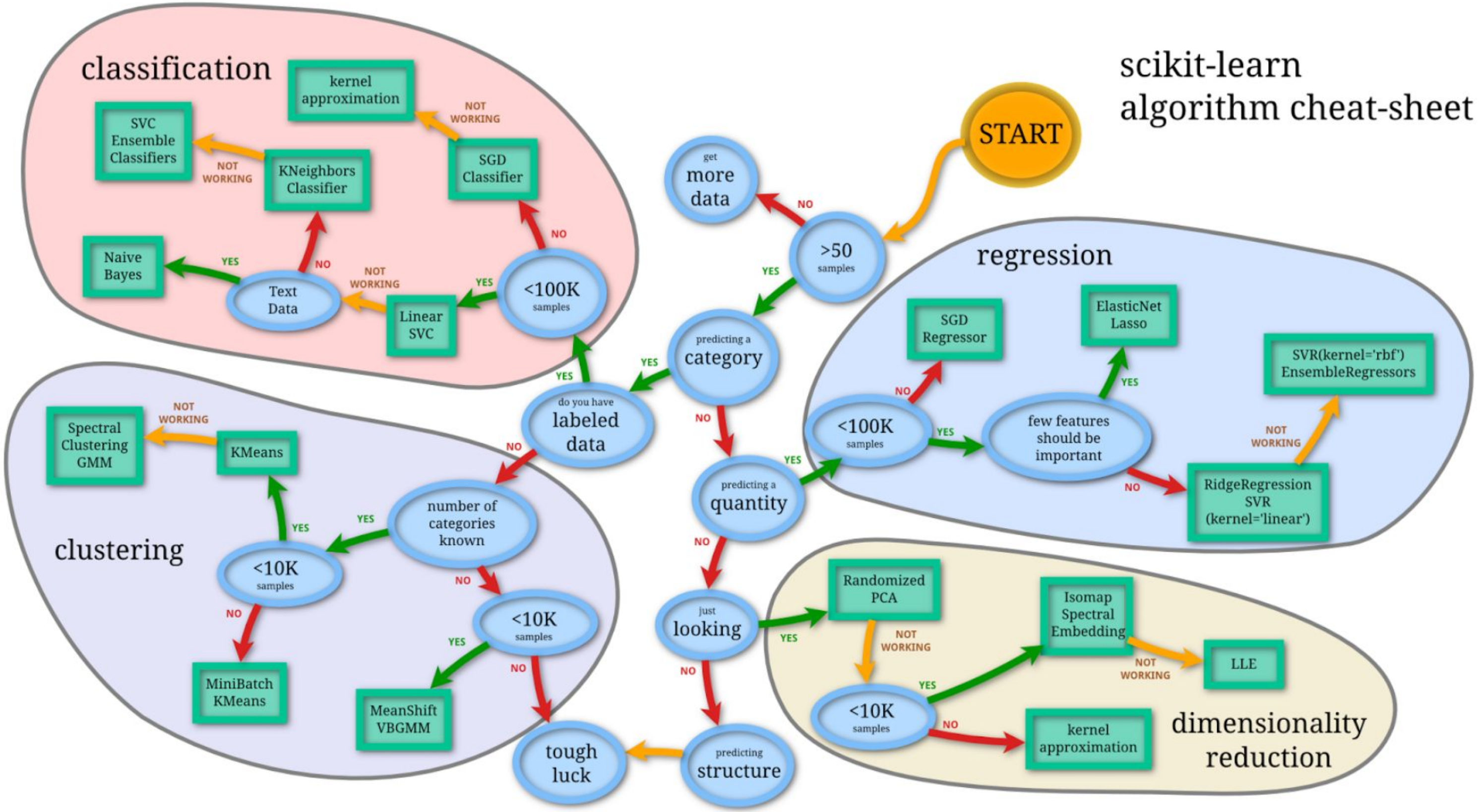
"Machine learning is the science of getting computers to act without being explicitly programmed."

by Andrew Ng
(probably inspired by Arthur Samuels)

… and also this

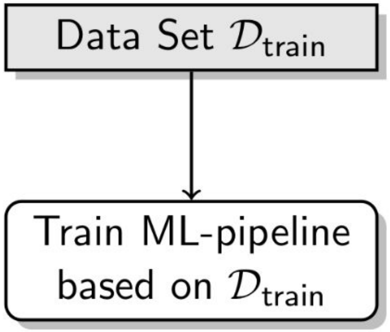
source: [XKDC]

Design Decisions

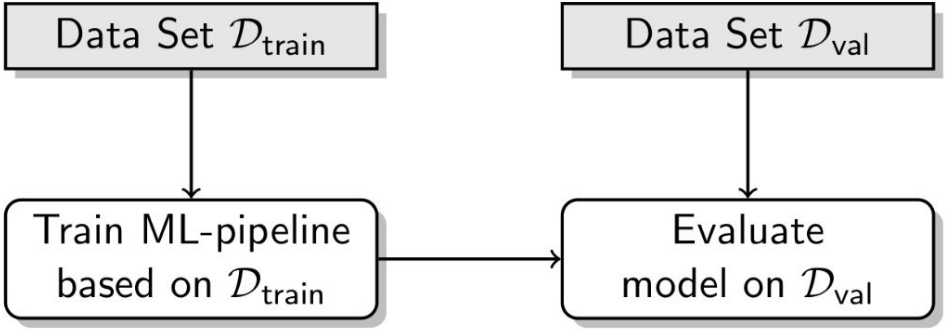


source: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

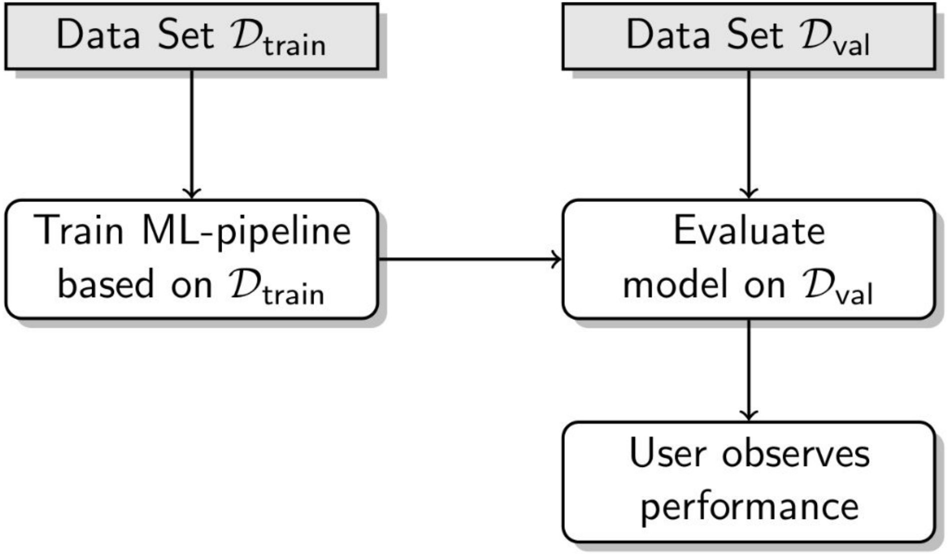
Manual Machine Learning Workflow



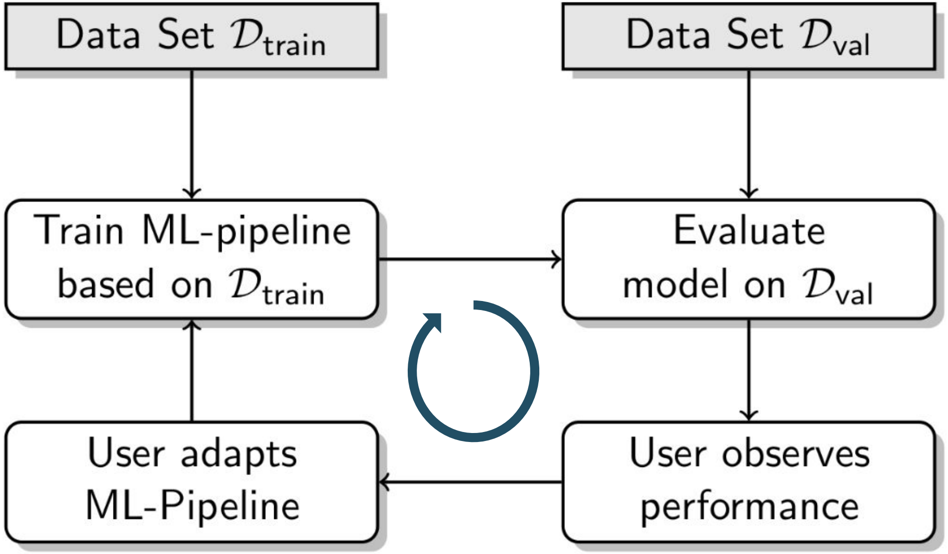
Manual Machine Learning Workflow



Manual Machine Learning Workflow



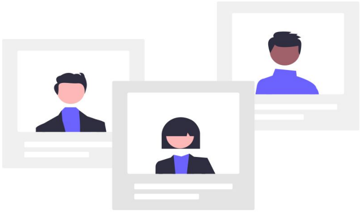
Manual Machine Learning Workflow



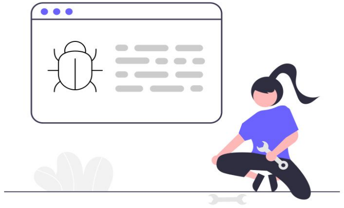
Challenges in Applying AI/ML these days

Required expertise in ML and AI

Long development time for new AI applications

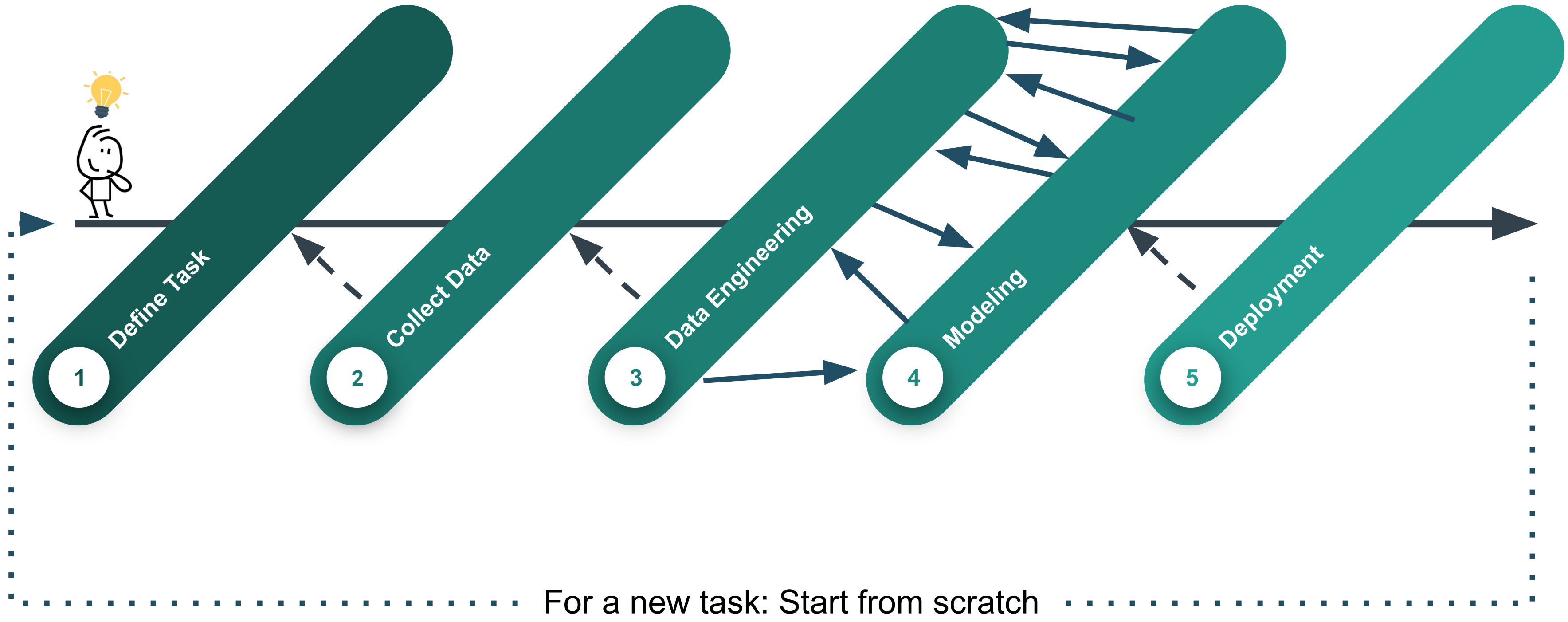


Few experts are available on the job market



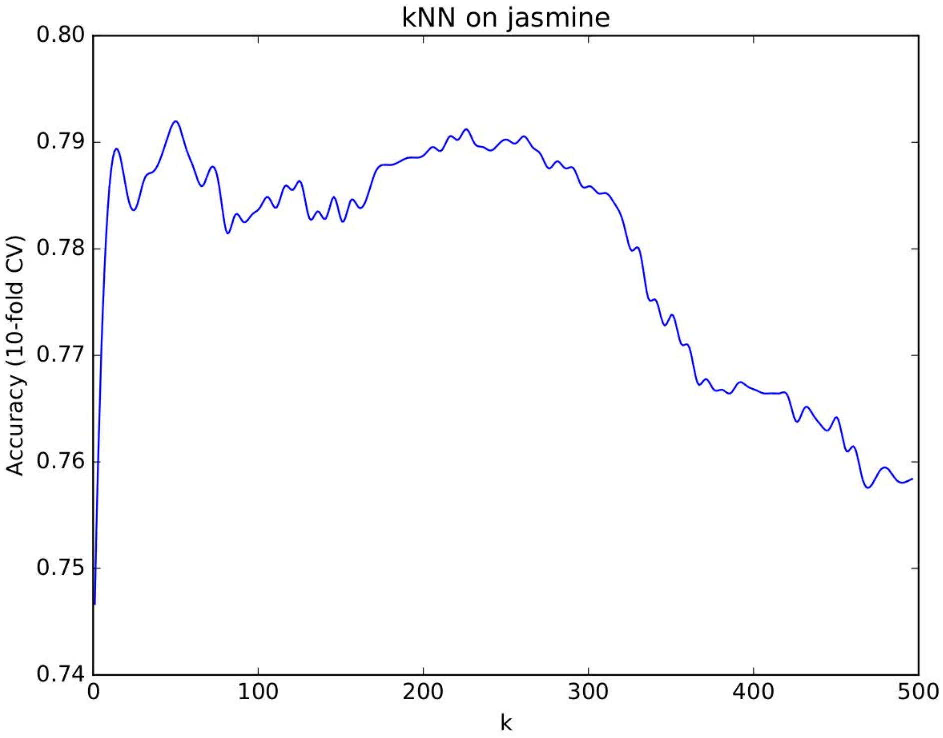
Unstructured and error-prone development of AI application

Why does ML development take a lot of time?



Toy Example: kNN

- *k*-nearest neighbors (kNN) is one of the **simplest ML algorithms**
- Size of neighbourhood (*k*) **is very important for its performance**
- The performance function depending on *k* is **quite complex** (not at all convex)



Goals of AutoML

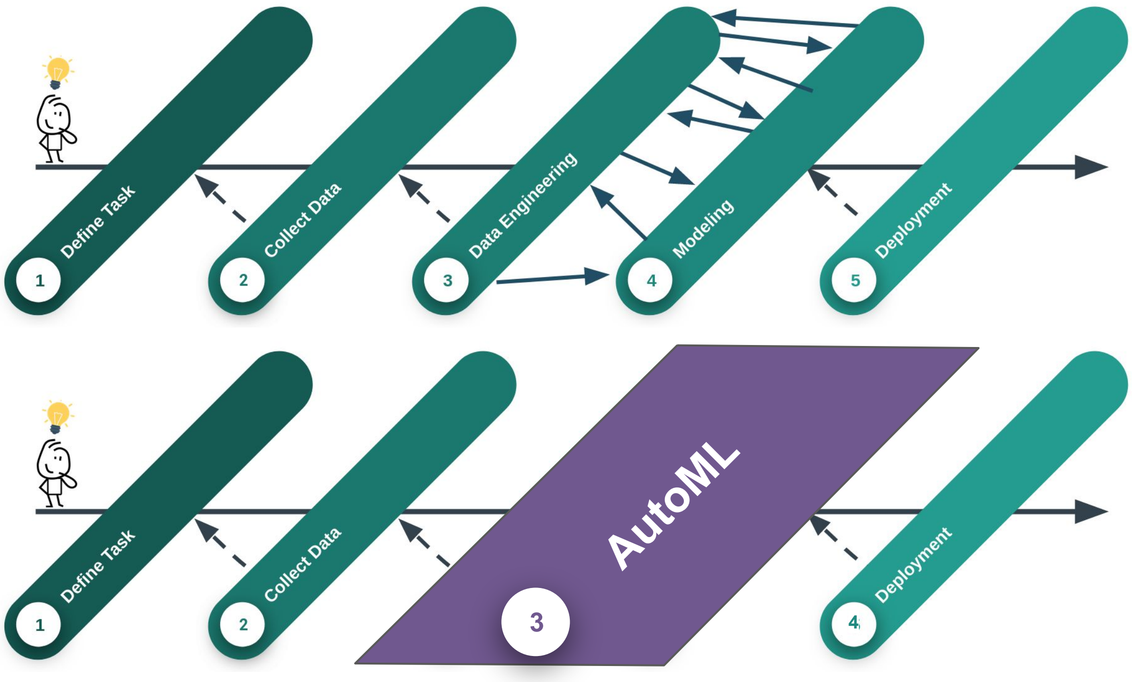
Goal: Progressively automate all parts of machine learning (as needed) to support users efficiently building their ML-applications.

Informal Definition: AutoML System

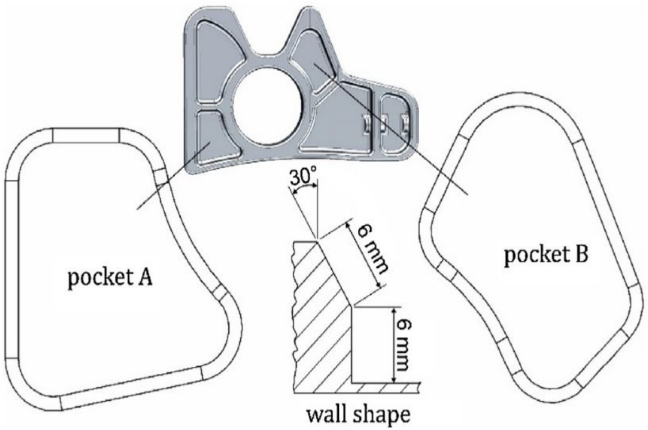
Given
- A **dataset**,
- a **task** (e.g. supervised classification),
- a **cost metric** (e.g., accuracy or RMSE),
- (optional) a **budget**

an AutoML System automatically determines the approach that performs best for this application.

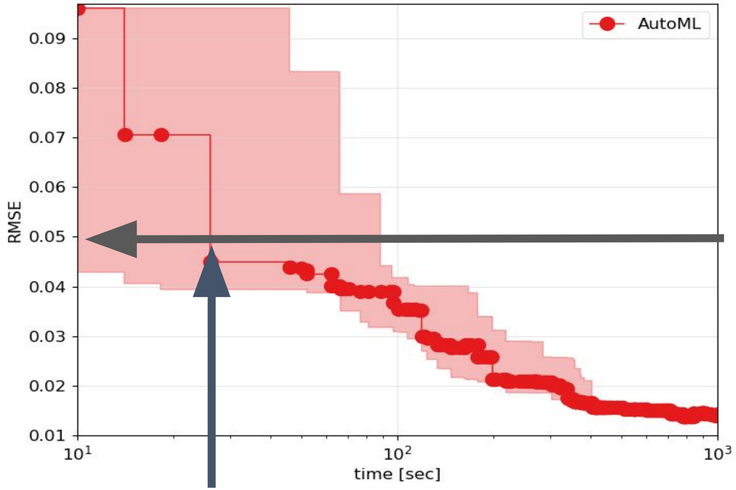
ML vs AutoML


Define Task
1
Collect Data
2
Data Engineering
3
Modeling
4
Deployment
5
Define Task
1
Collect Data
2
AutoML
3
Deployment
4

Motivating Example: Shape Error Prediction in Milling Process



[Denkena et al. 2020]



State of the art by human domain expert

Outperforming human domain expert after ~30sec
(+ some time to write a parser for the data)

Advantages

AutoML enables

- 🚀 More **efficient** research (and development of ML applications)
  → AutoML has been shown to outperform humans on subproblems
- 🧮 More **systematic** research (and development of ML applications)
  → no (human) bias or unsystematic evaluation
- 📋 More **reproducible** research
  → since it is systematic!
- 🦾 **Broader use** of ML methods
  → less required ML expert knowledge
  → not only limited to computer scientists

Challenges

But, it is not that easy, because

🔁 Each dataset potentially requires **different optimal ML-designs**

→ Design decisions have to be made for each dataset again

⌛ Training of a single ML model can be **quite expensive**

→We can not try many configurations

❓Mathematical **relation** between design and performance is (often) **unknown**

→ Gradient-based optimization not easily possible

🎛️ 🎚️ Optimization in **highly complex spaces**

→ including categorical, continuous and conditional dependencies

Building a Successful AutoML Tool

→ There is not a single way for building successful AutoML tools,
but there are well established approaches.

- Hyperparameter optimization
- Neural architecture search
- Full ML pipeline search
- human-centered developement
- (Dynamic approaches)

→ But how can we combine all of these?

The Team

>> Who are we and you?

Marius Lindauer

Professor at the Leibniz University of Hannover (Germany)

Head of Institute of AI &
Head of the machine learning group

Won 1st and 2nd international AutoML challenge; ERC Starting Grant for ixAutoML

Co-author of popular AutoML tools Auto-sklearn, Auto-PyTorch, SMAC3

Co-Head of automl.org

Co-Founder and advisory board member of the research network COSEAL (Configuration and Selection of Algorithms)

Katharina Eggensperger

Early Career Group Leader at the **University of Tübingen (Germany)** for **AutoML for Science**

Won 1st and 2nd **international AutoML challenge** (this was a team effort!)

Core developer of popular AutoML tools, **Auto-sklearn** and **SMAC**

Junior-Head of automl.org

And you?

https://tinyurl.com/automl-miro

For what are you using ML?
Have you ever heard of AutoML?
Have you ever used AutoML?

Where are you from?
Do you look for others to network?

… and what you will learn this week

```
>> Finally!
```

Learning Goals

After this week you:

- Know what **HPO, NAS, BO** and **CASH** is
- Can **discuss** about and **apply** AutoML methods
- Know **how to tune hyperparameters**
- Know what **neural architecture search** is and how to use it
- Combine the best of both worlds: Systematic AutoML and human expertise in **human-centered AutoML**
- Know about **AutoML systems and their limitations**
- Want to use AutoML for your next ML project ;-)

The Challenges

>> Is it really that complicated?

Hyperparameters

scikit learn

Home Installation Documentation Examples

Google Custom Search

Previous sklearn.svm. O... | Next sklearn.svm. SVR | Up API Reference

scikit-learn v0.20.3
Other versions

Please **cite us** if you use the software.

sklearn.svm.SVC
Examples using sklearn.svm.SVC

sklearn.svm.SVC

class sklearn.svm. **SVC** (*C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None*) [source]

C-Support Vector Classification.

The implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples.

The multiclass support is handled according to a one-vs-one scheme.

For details on the precise mathematical formulation of the provided kernel functions and how gamma, coef0 and degree affect each other, see the corresponding section in the narrative documentation: Kernel functions.

Read more in the User Guide.

Parameters:

C : *float, optional (default=1.0)*
Penalty parameter C of the error term.

kernel : *string, optional (default='rbf')*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, optional (default=3)*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *float, optional (default='auto')*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

Current default is 'auto' which uses 1 / n_features, if `gamma='scale'` is passed then it uses 1 / (n_features * X.var()) as value of gamma. The current default of gamma, 'auto', will change to 'scale' in version 0.22. 'auto_deprecated', a deprecated version of 'auto' is used as a default indicating that no explicit value of gamma was passed.

coef0 : *float, optional (default=0.0)*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : *boolean, optional (default=True)*
Whether to use the shrinking heuristic.

probability : *boolean, optional (default=False)*
Whether to enable probability estimates. This must be enabled prior to calling fit, and will slow

PyTorch

SGD

CLASS torch.optim.SGD(*params, lr=<required parameter>, momentum=0, dampening=0, weight_decay=0, nesterov=False, *, maximize=False, foreach=None, differentiable=False*) [SOURCE]

Implements stochastic gradient descent (optionally with momentum).

Parameters:

- **params** (*iterable*) – iterable of parameters to optimize or dicts defining parameter groups
- **lr** (*float*) – learning rate
- **momentum** (*float, optional*) – momentum factor (default: 0)
- **weight_decay** (*float, optional*) – weight decay (L2 penalty) (default: 0)
- **dampening** (*float, optional*) – dampening for momentum (default: 0)
- **nesterov** (*bool, optional*) – enables Nesterov momentum (default: False)
- **maximize** (*bool, optional*) – maximize the params based on the objective, instead of minimizing (default: False)
- **foreach** (*bool, optional*) – whether foreach implementation of optimizer is used. If unspecified by the user (so foreach is None), we will try to use foreach over the for-loop implementation on CUDA, since it is usually significantly more performant. (default: None)
- **differentiable** (*bool, optional*) – whether autograd should occur through the optimizer step in training. Otherwise, the step() function runs in a torch.no_grad() context. Setting to True can impair performance, so leave it False if you don't intend to run autograd through this instance (default: False)

HPO: Hyperparameter Optimization

Definition

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of an ML algorithm $\mathcal{A}$ with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{\text{train}}$ and evaluated on $\mathcal{D}_{\text{val}}$.

The *hyper-parameter optimization (HPO)* problem is to find a hyper-parameter configuration that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remarks:

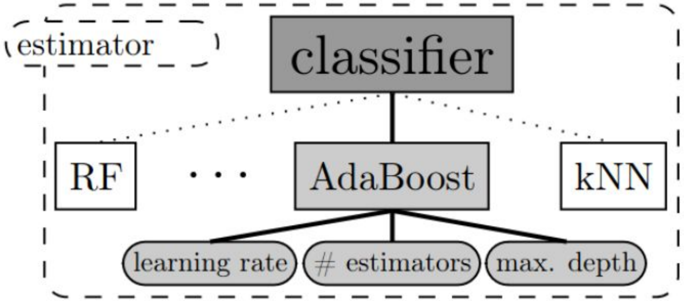
- $\arg\min$ returns a set of optimal points of a given function. It suffices to find one element of this set and thus we use $\in$ instead of $=$.
- Sometimes, we want to optimize for different metrics, instead of one
  $\rightsquigarrow$ multi-objective optimization and Pareto fronts

Lecture 2

Choosing an Algorithm

- Many ML-algorithms exist
- Most of these (still) have a reason for existence
- Examples for classification:
  - logistic regression
  - k-nearest neighbor
  - naïve Bayes
  - random forest
  - gradient boosting
  - multi-layer perceptron
  - SVM
  - decision tree
  - residual networks
- [Fernández-Delgado et al. 2014] studied 179 classifiers on 121 datasets

→ In practice: We want to jointly choose the best ML-algorithm **and** its hyperparameters



CASH: Combined Algorithm Selection and Hyperparameter Optimization

[Thornton et al. 2013]

Definition

Let

- $\mathbf{A} = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_k\}$ be a set of algorithms (a.k.a. portfolio)
- $\mathbf{\Lambda}$ be a set of hyperparameters of each machine learning algorithm $\mathcal{A}_i$
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

we want to find the best combination of algorithm $\mathcal{A} \in \mathbf{A}$ and its hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ minimizing:

$$(\mathcal{A}^*, \boldsymbol{\lambda}^*) \in \operatorname*{arg\,min}_{\mathcal{A} \in \mathbf{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}} c(\mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Please don't trust LLMs telling you that AutoML was invented by Google in 2017. Obviously wrong!

Lecture 2

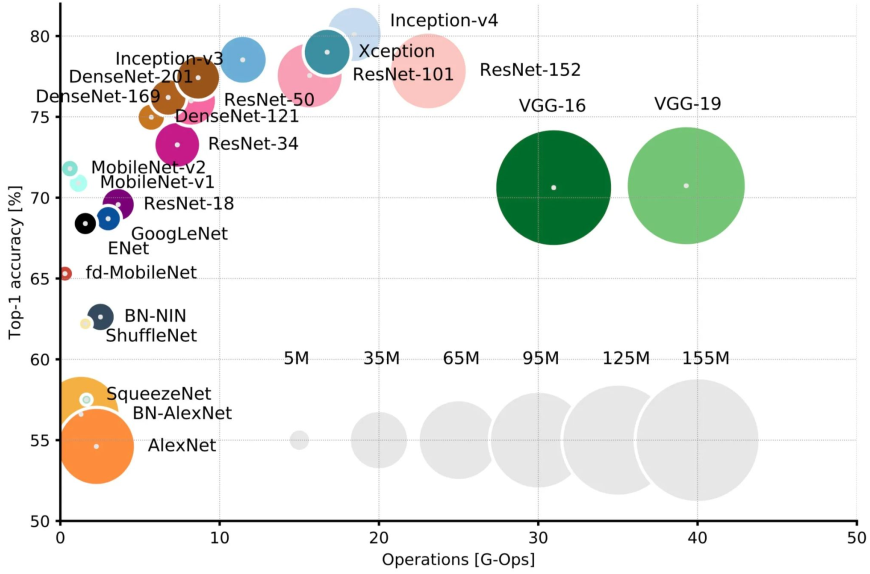
Architectures of Neural Networks

Many architectures exist and differ in

- Depth
- Resolution
- Width
- Operators
- Connections
- ...

Already on a single dataset (e.g. ImageNet), it is **not obvious** which architecture to choose

- On **different** datasets → different architectures
- On **similar** datasets → scaled versions of known architectures (e.g. ImageNet and Cifar10)



Source: [Culurciello et al. 2018]

NAS: Neural Architecture Search

Definition

Let

- $\boldsymbol{\lambda}$ be an architecture for a deep neural network N with domain $\boldsymbol{\Lambda}$,
- $\mathcal{D}_{opt}$ be a dataset which is split into $\mathcal{D}_{train}$ and $\mathcal{D}_{valid}$
- $c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ denote the cost of $N_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and evaluated on $\mathcal{D}_{valid}$.

The *neural architecture search (NAS)* problem is to find an architecture that minimizes this cost:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} c(N_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Remarks:

- very similar to the HPO definition
- In practice, you want jointly optimize HPO and NAS [Zela et al. 2018]

Lecture 3

Summary

HPO Search for the best hyperparameter configuration of a ML algorithm

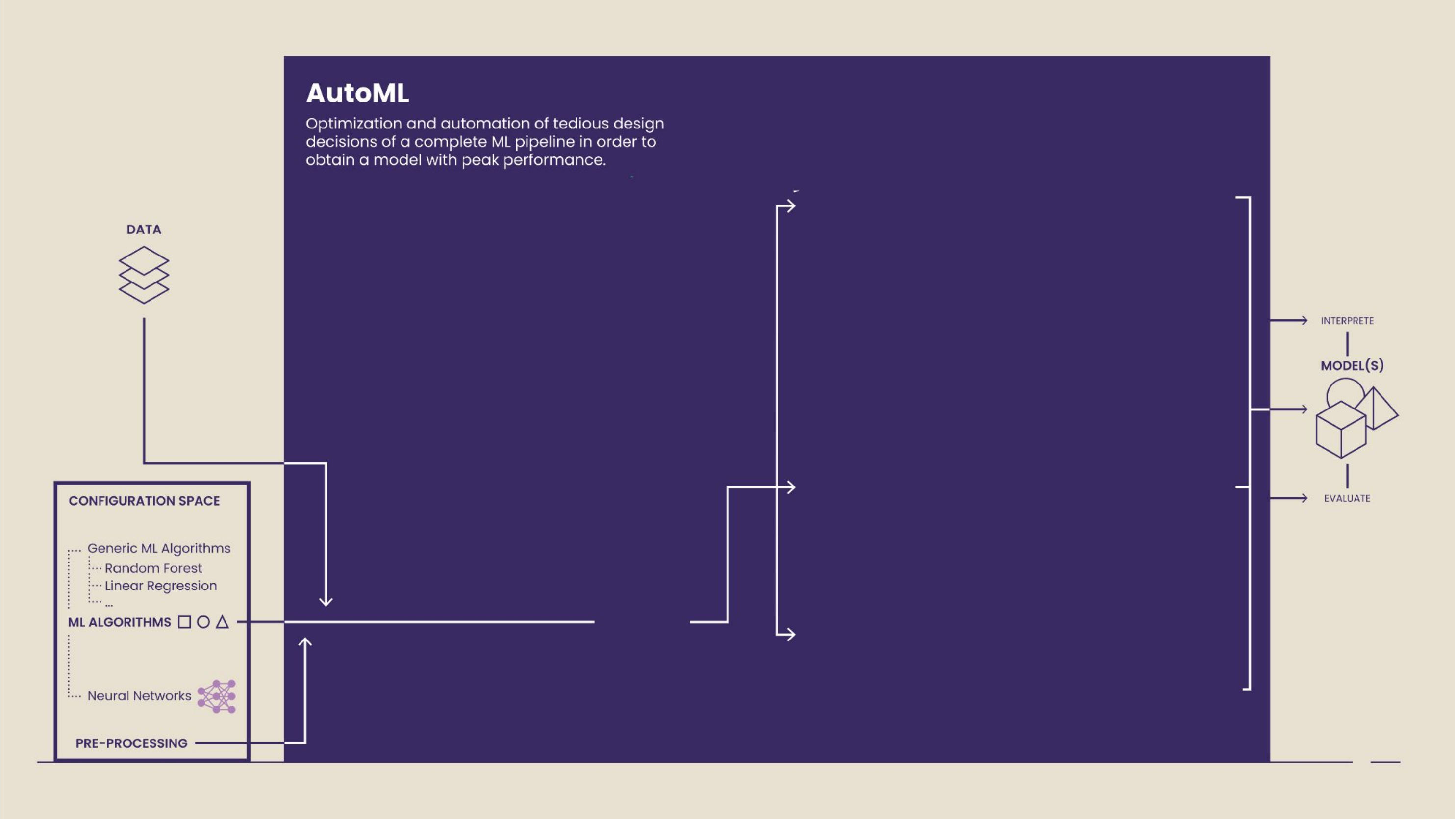
CASH Search for the best combination of algorithm and hyperparameter configuration

NAS Search for the architecture of neural network

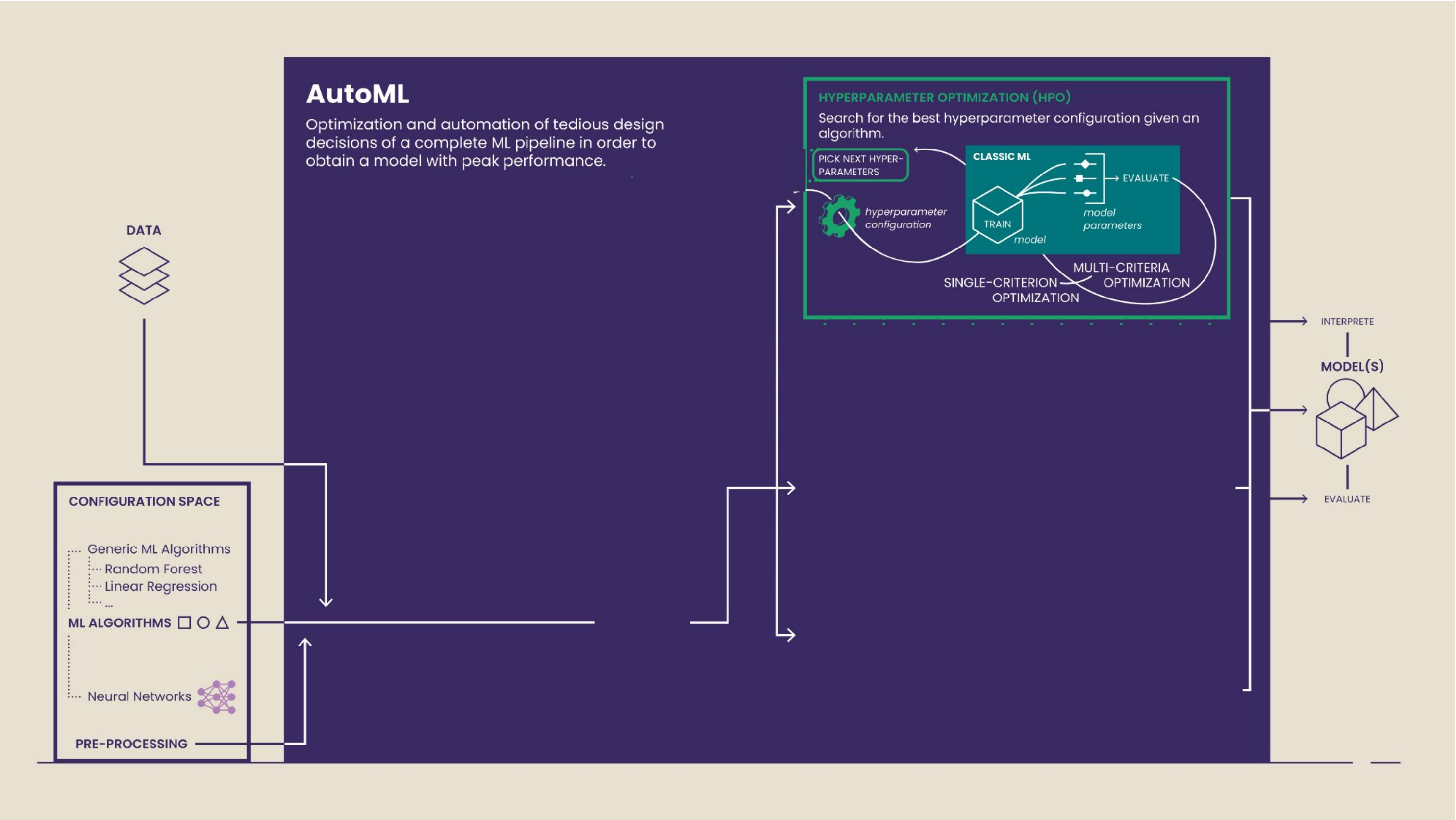
The Big Picture II

```
>> Show me a nice picture!
```

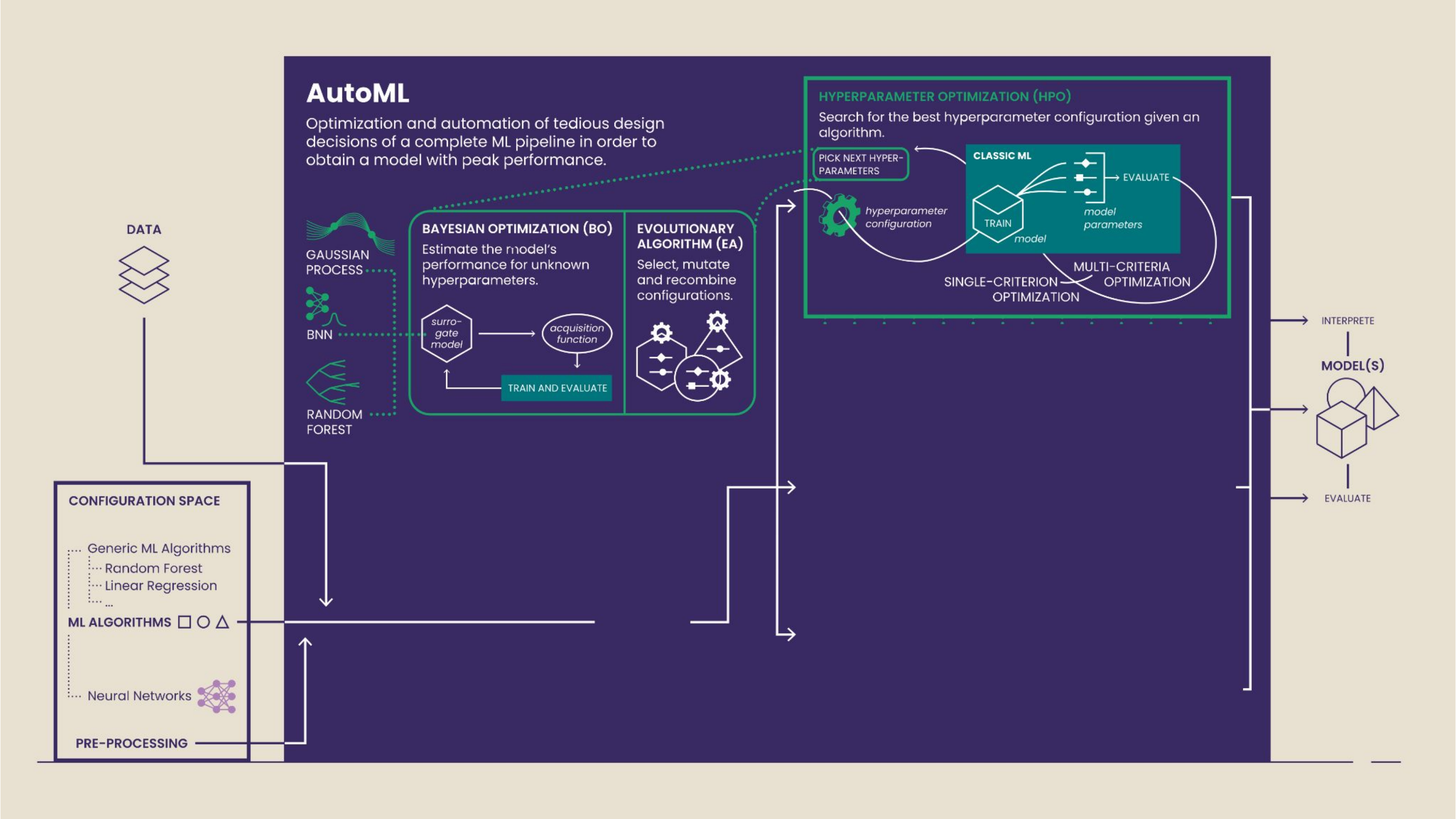
How everything is connected



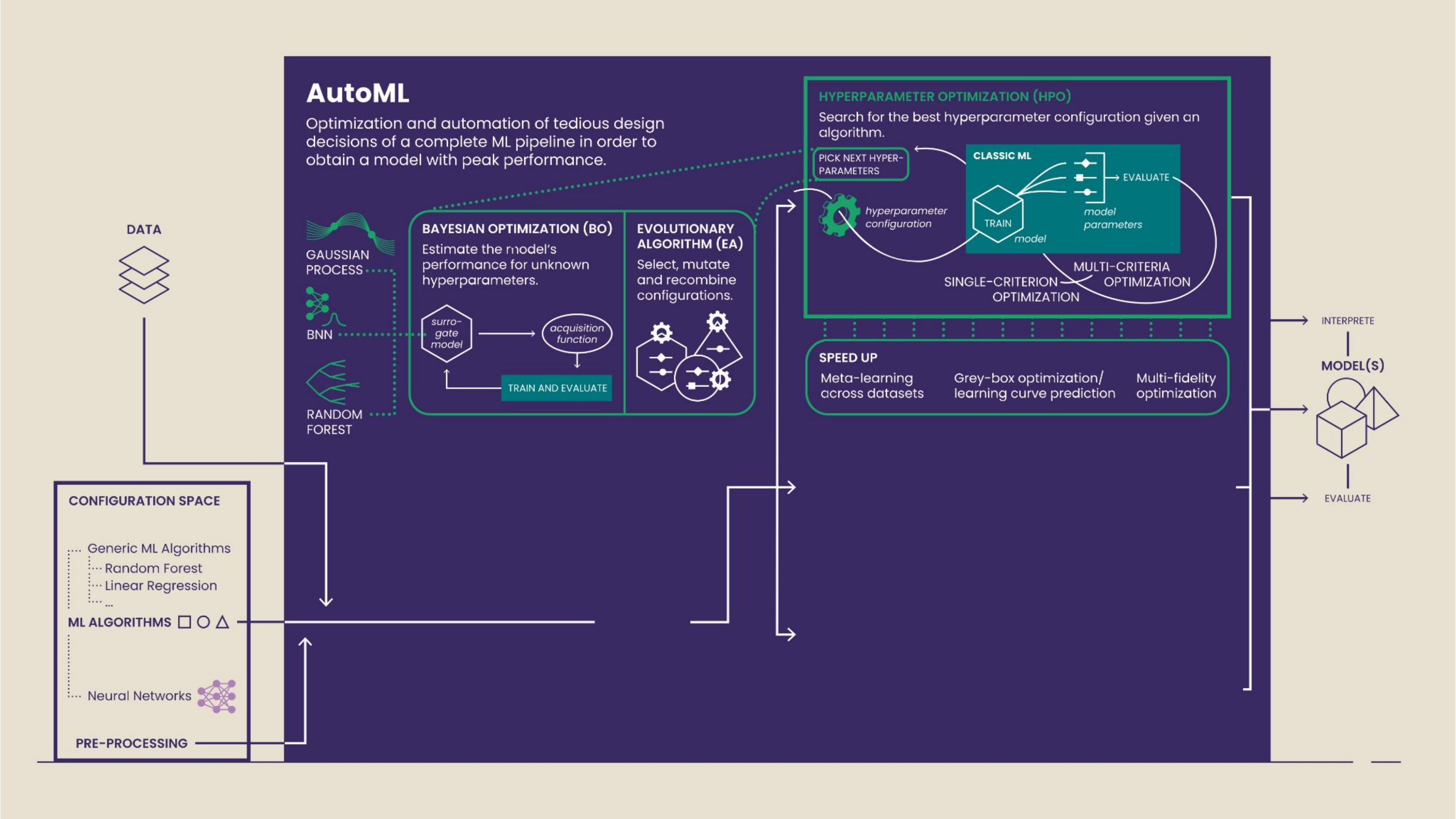
How everything is connected



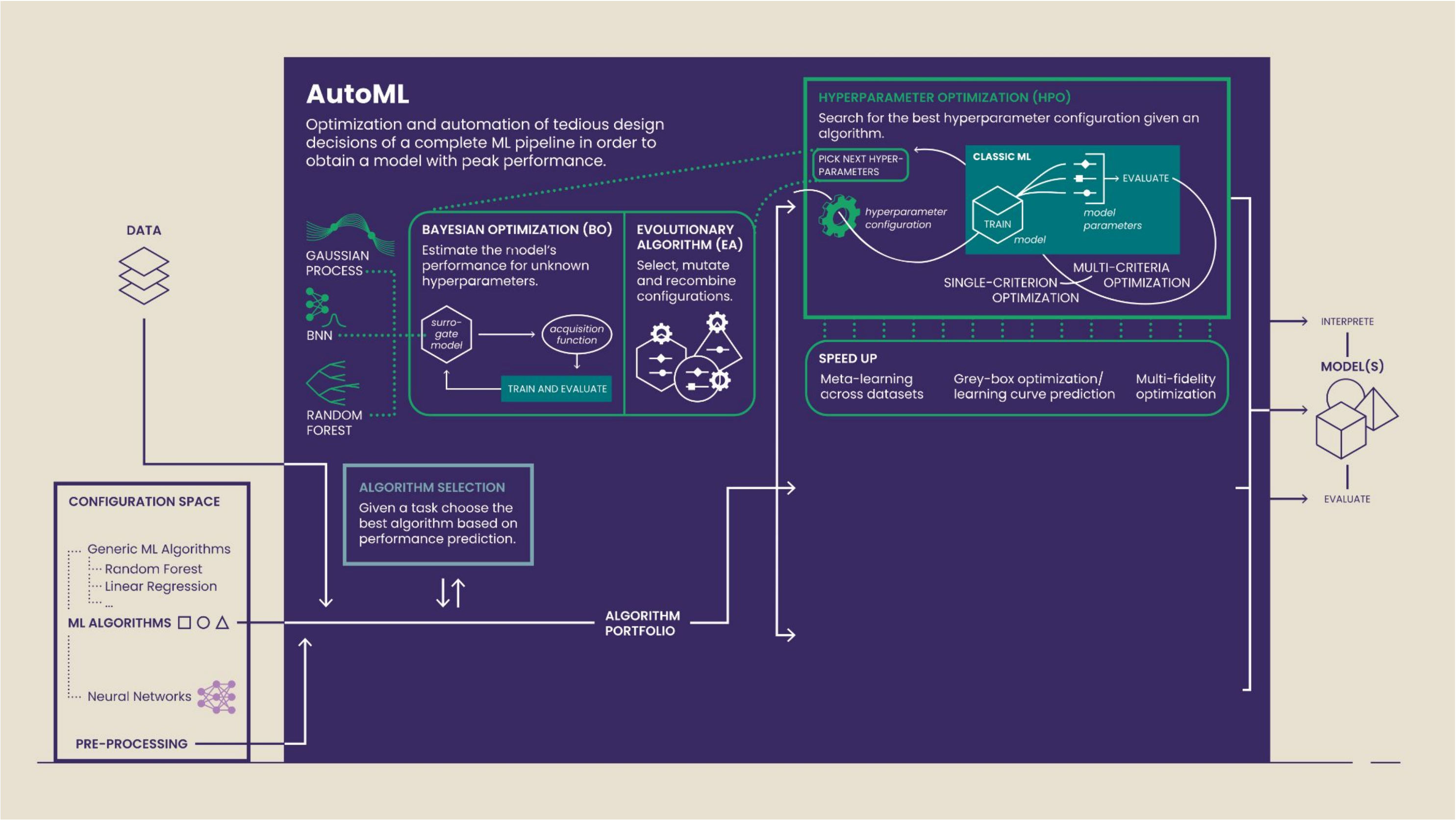
How everything is connected



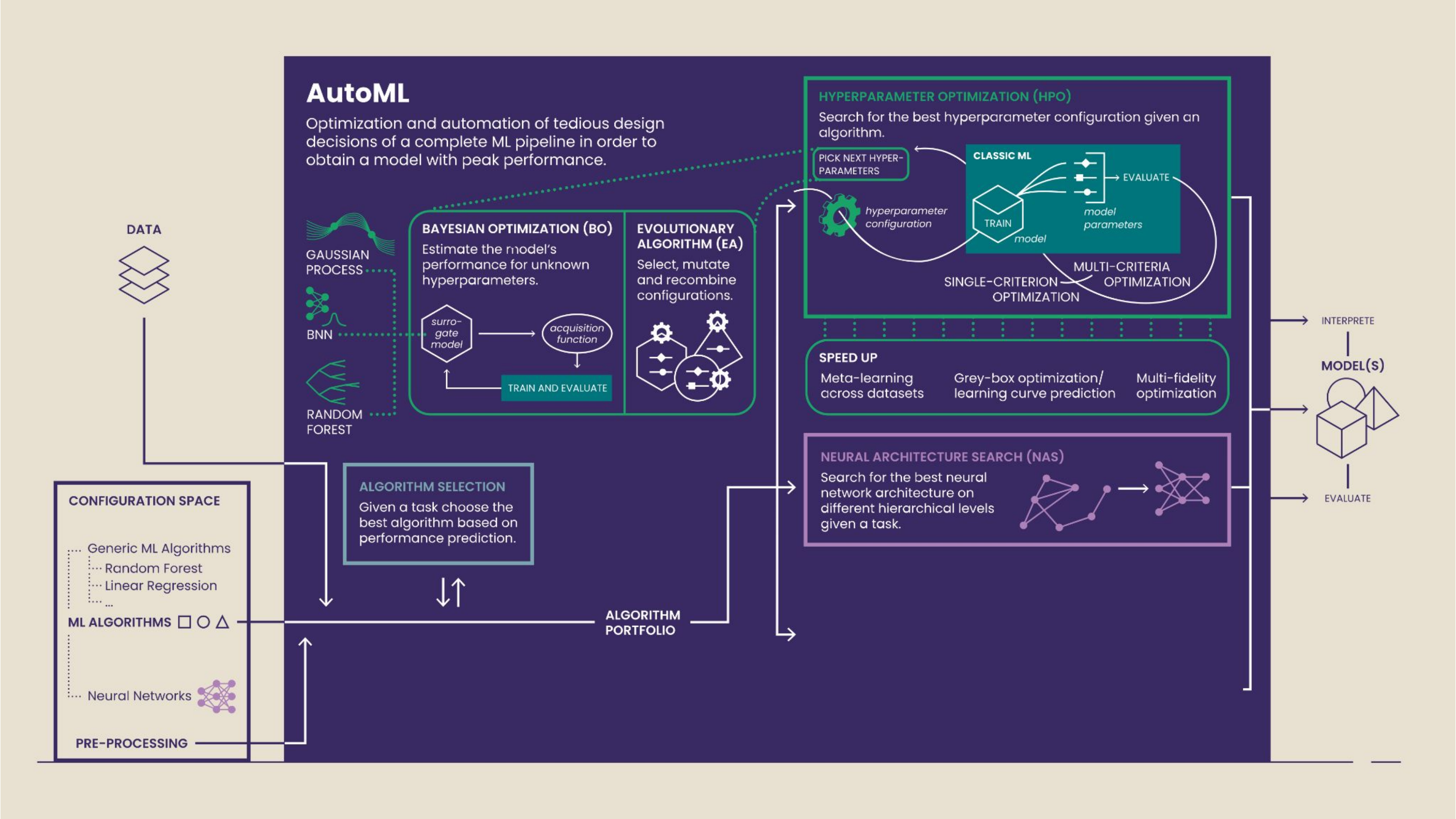
How everything is connected



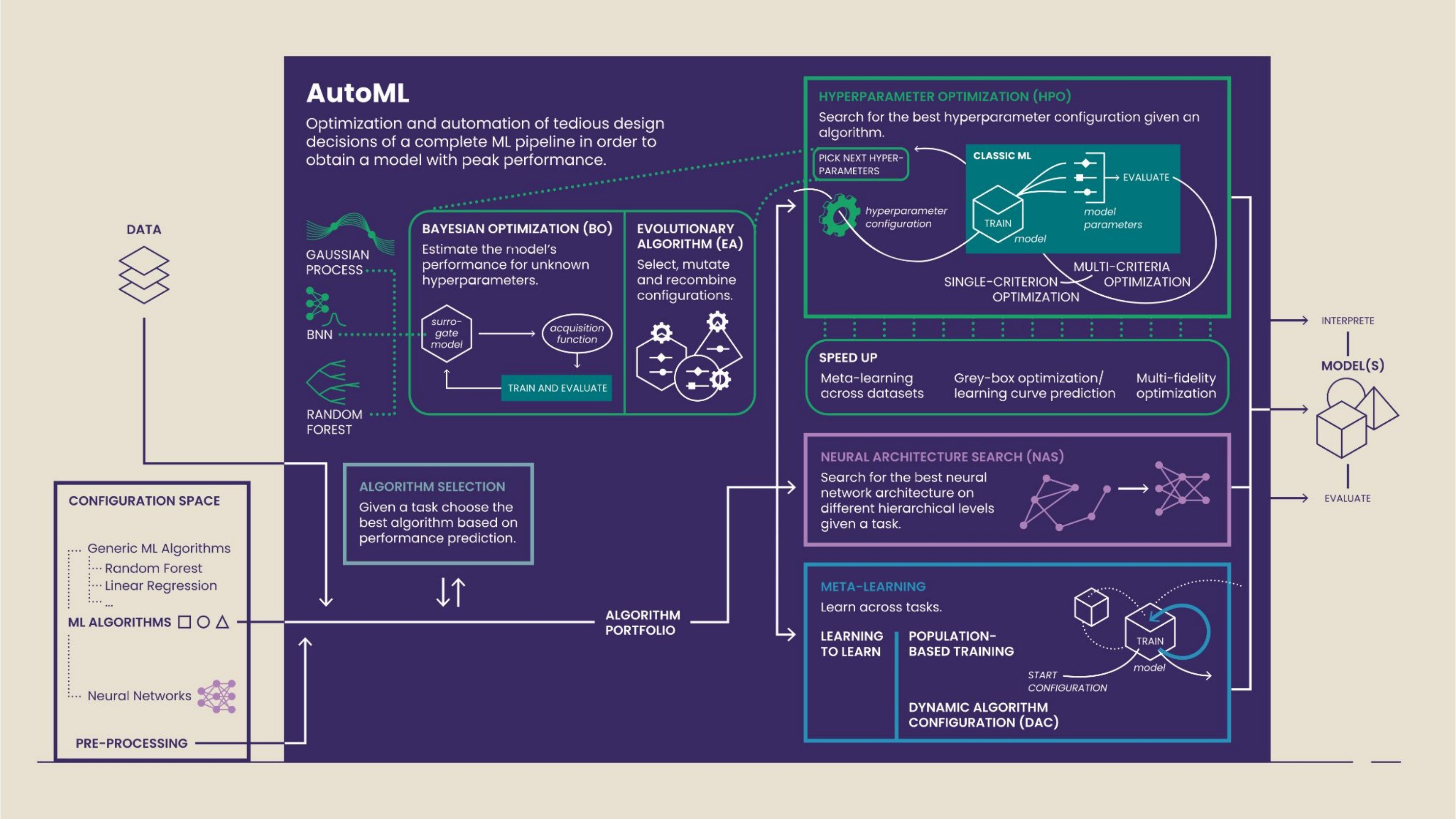
How everything is connected



How everything is connected



How everything is connected



The Risks

`>> Anything else I should know?`

Risks of AutoML

1. Users **blindly apply** AutoML.
   → Users might wonder why (Auto-)ML does not work after they passed in *poor data*.
2. **Automation Bias**.
   → Users might not use human reasoning skills and do not second guess machine decisions
3. Non-ML experts **can use ML without knowing** the risks and consequences of ML itself.
   → E.g., bias in data and trained models

Might lead to:

- **inaccurate** models due to lack of understanding of statistical concepts
- **biased** and **unfair** models due to lack of understanding ethical practices
  - see also discussion on whether fairness can be automated [Weerts et al. 2023]

TODO for AutoML: Raise Awareness of these risks!

Lecture 5

Questions?

Kahoot Quiz !

Evaluating ML Models and AutoML

`>> Okay, what’s the first step?`

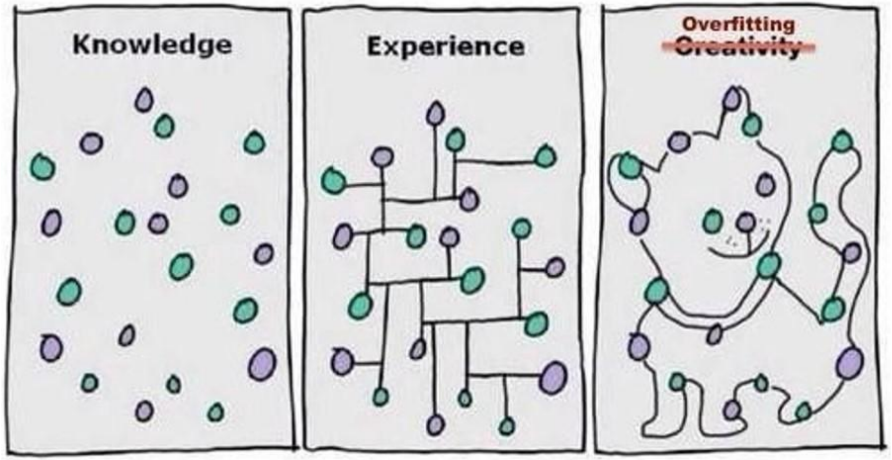
What do we want?

We want solutions that **generalize to new data!**

→ “reasonable” predictions on **new data**

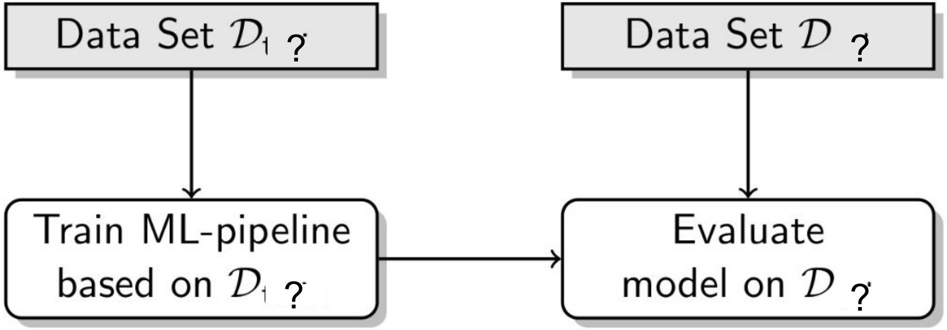
This might include:

- ignoring outliers
- smooth
- capturing general trend



Source: Kaushik, 2016

Basic ML Evaluation



Basic ML Evaluation

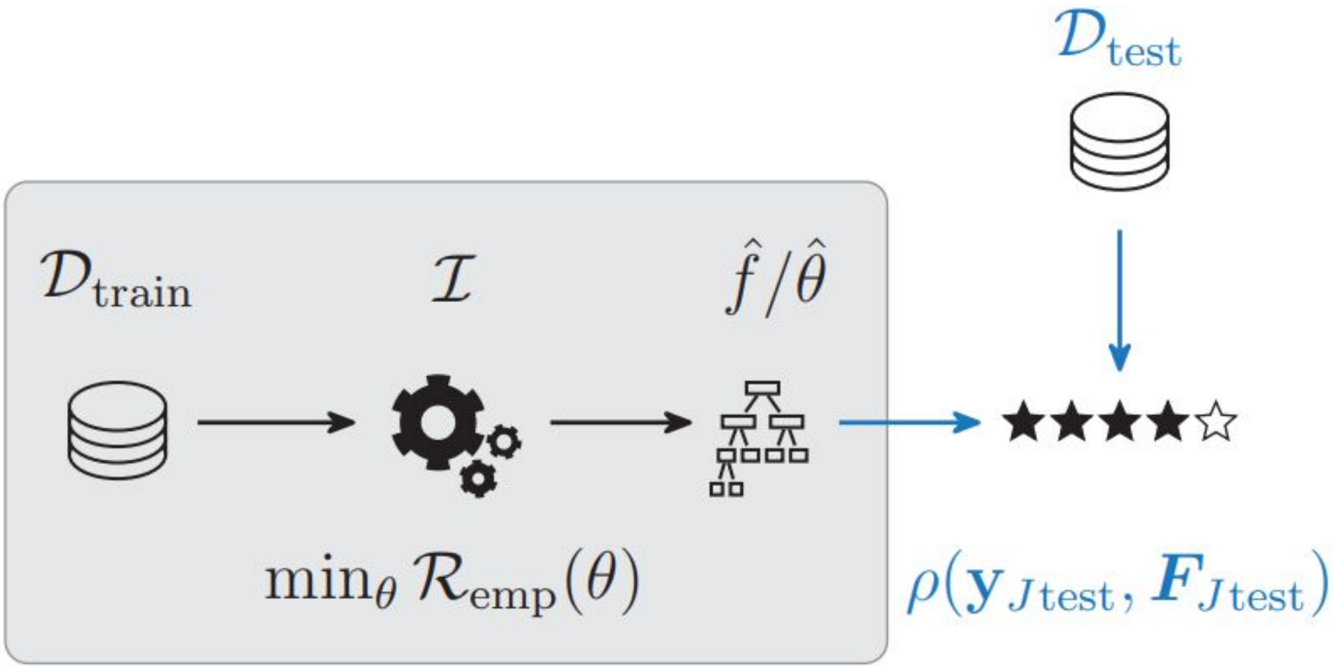
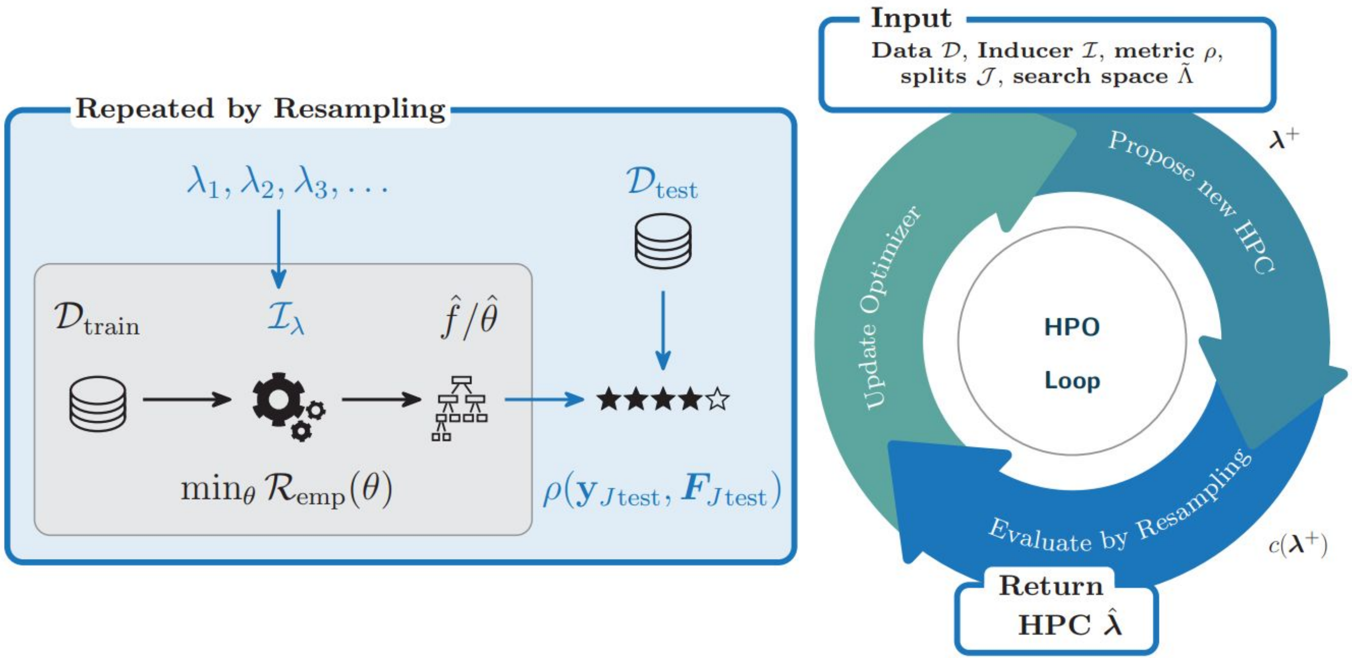


Image: [Bischl et al. 2023]

General HPO Loop



Warning: If you do this on your holdout test data, you only get a biased cost estimate!

Image: Bischl et al. 2023

Nested CV for AutoML

For ML users

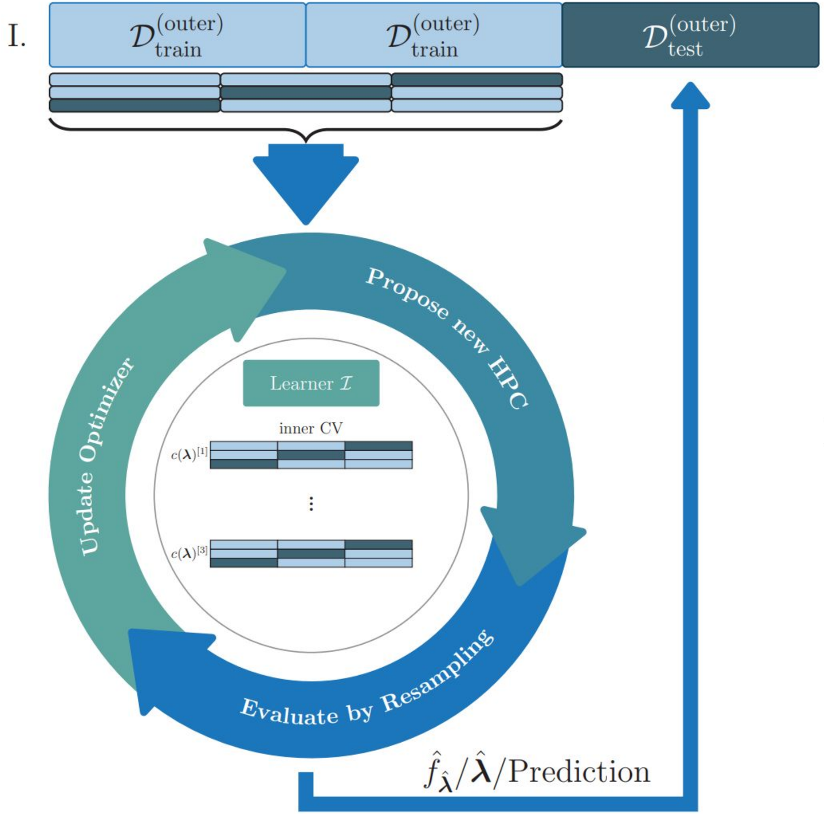


Image: Bischl et al. 2023

Nested CV for AutoML

For ML users

For AutoML research

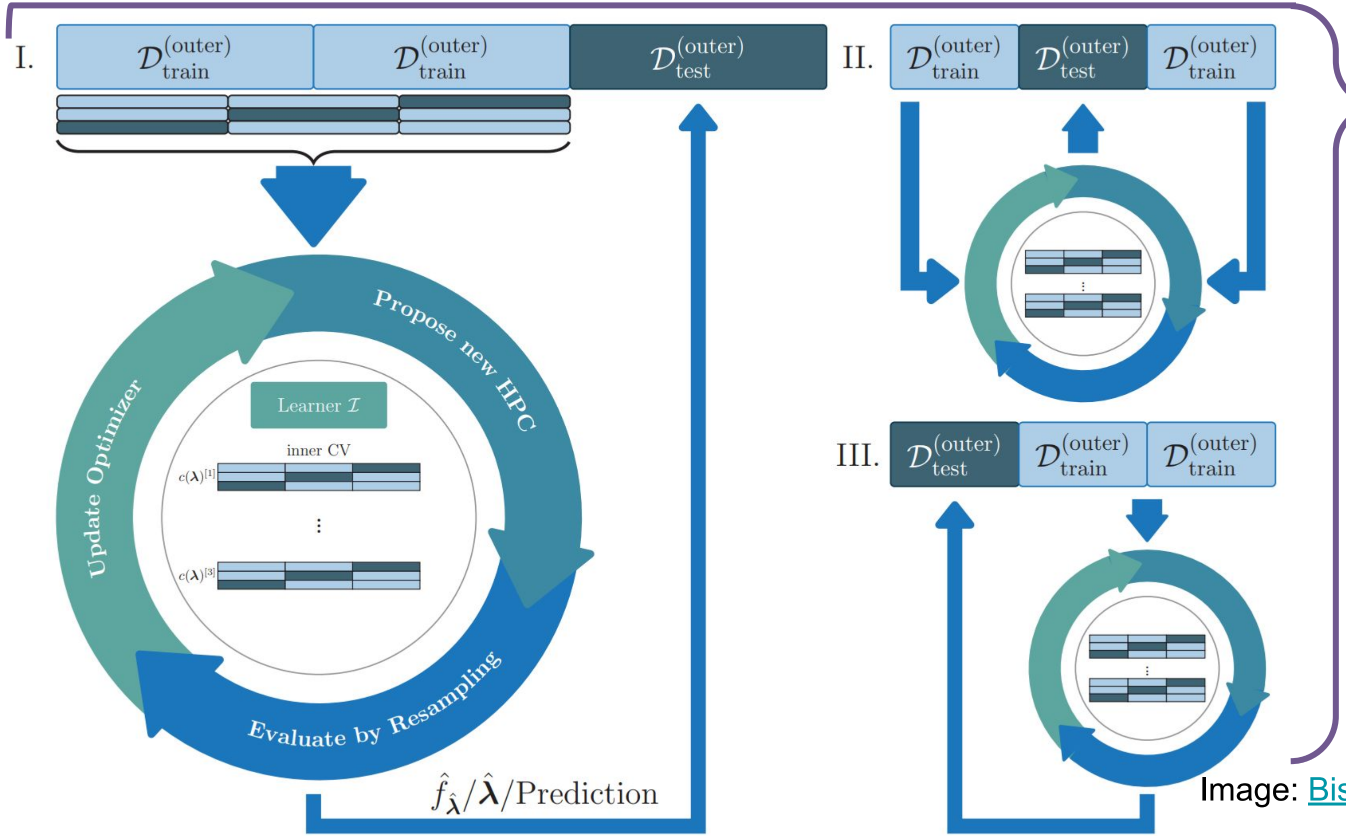


Image: Bischl et al. 2023

Illustration: Resampling vs. Nested Resampling

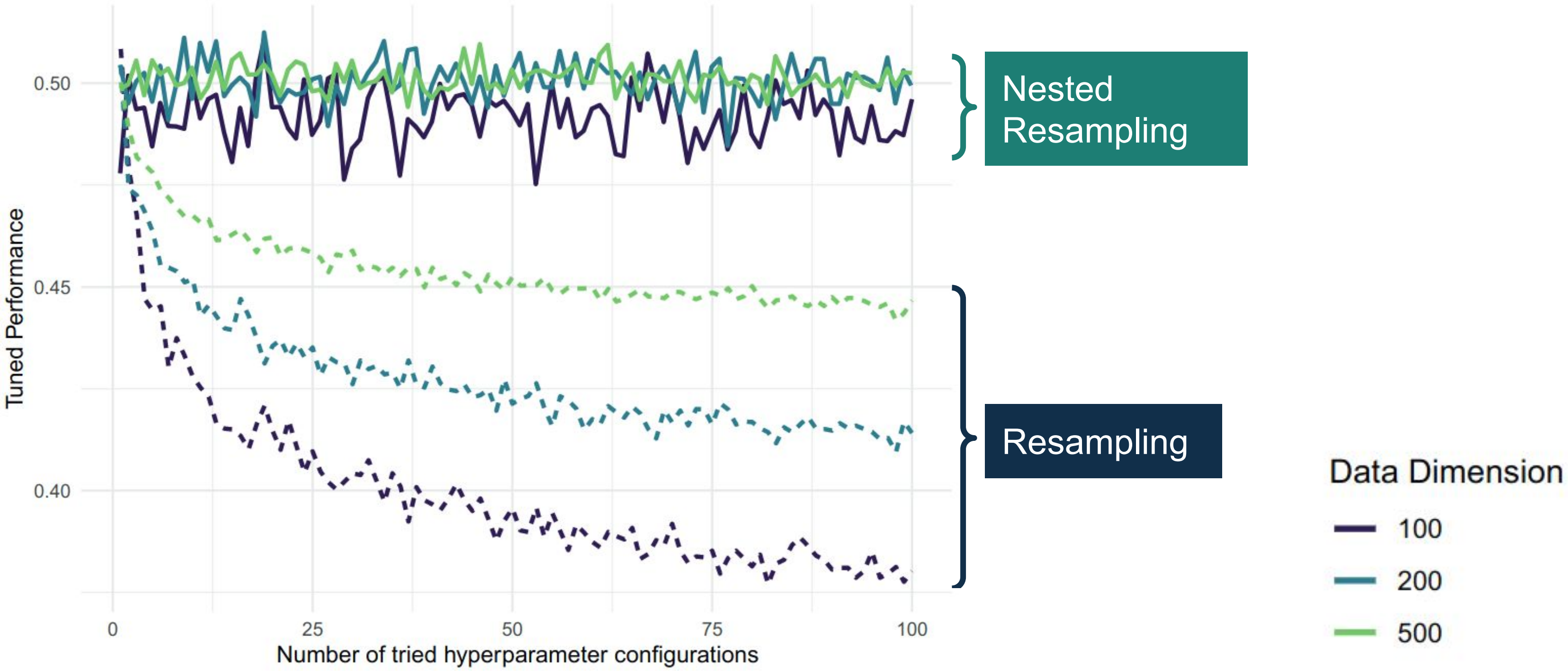


Image: [Bischl et al. 2023]

How to Evaluate AutoML Methods …

If we define AutoML as an optimization process, the incumbent solution (i.e., the best found configuration so far) gradually improves over time

How long will a user run the AutoML process? *

☕ Coffee break (15min) | 🗨️ Meeting (1h) | 💤 Over night (16h) | 🤸 Over the weekend (48+h)

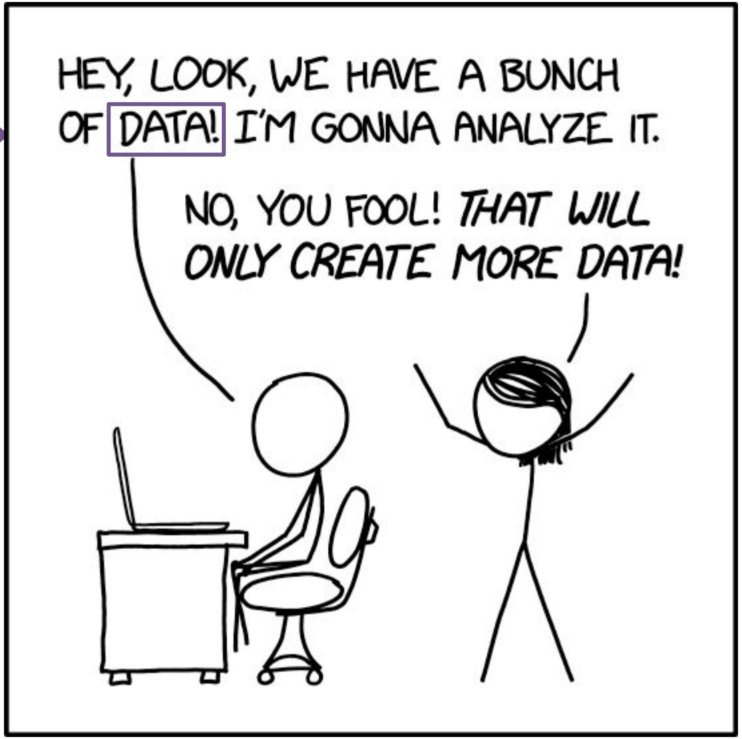
→ **Anytime performance is important!**

The AutoML tool should return the **best possible solution at each time point**

(*) Recent work on Bayesian Optimization provides a heuristic to stop [Makarova et al. 2022]

… and what to do with the results?

Several runs of our HPO tool SMAC



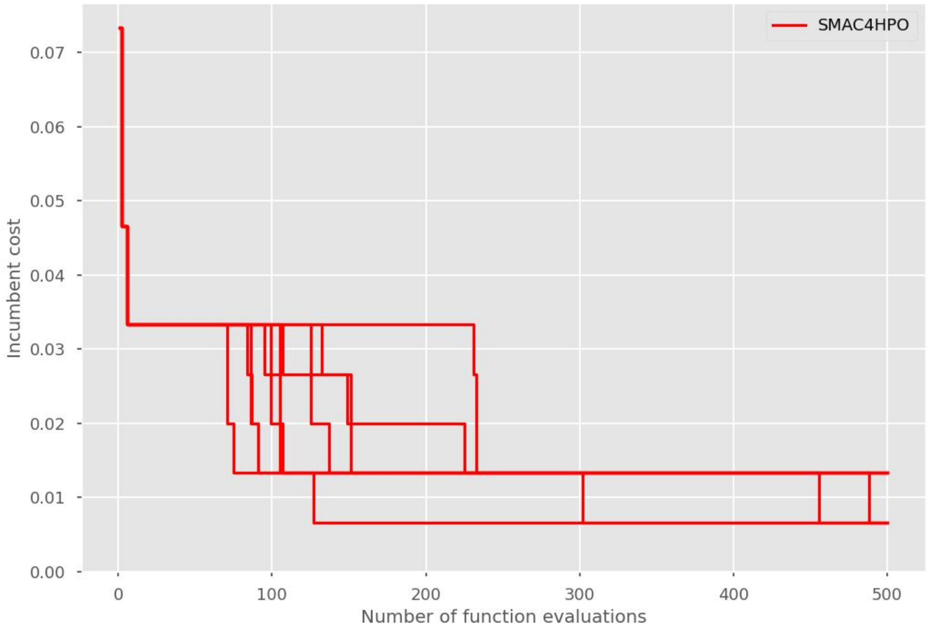
source: XKDC

Plotting over #evaluations

Several runs of our HPO tool SMAC over **number of function evaluations**

Insights:

- AutoML is very **noisy**
  → **Repeated measurements**!
- Incumbent configuration only **changes from time to time**
  → **Step function**!

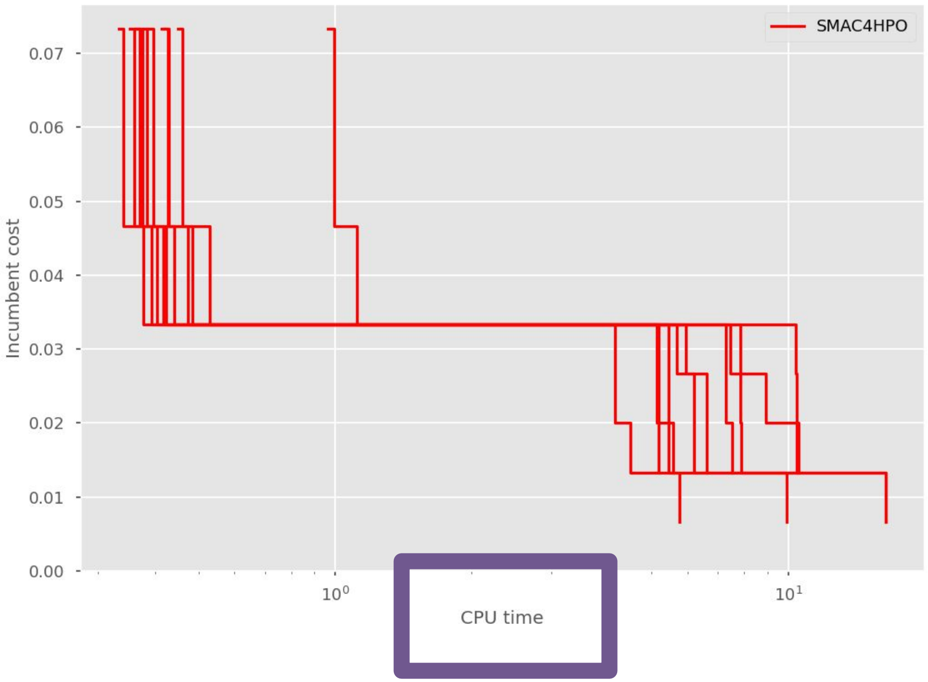


Plotting over Time

Several runs of our HPO tool SMAC over **CPU time**

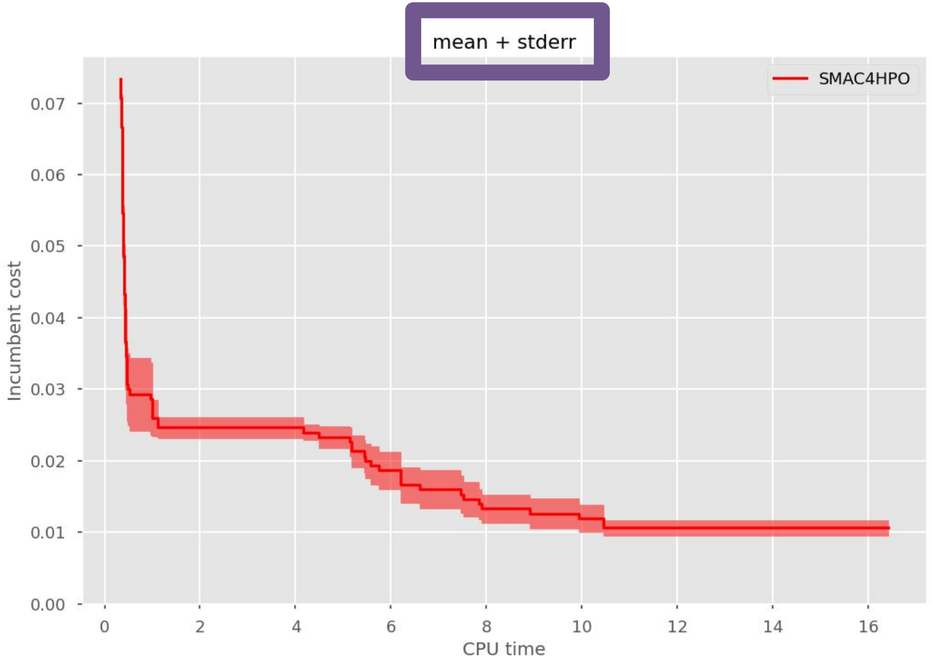
Insights:

- AutoML tools also induce **overhead** → Plotting over **CPU time** can be better
- Most improvements in the **beginning** → **x-axis** on **log**-scale
- (depending on the importance of **small improvements**, also **y-axis** on **log**-scale)



Mean and Standard Error

Mean across **several** runs of our HPO tool SMAC over **CPU time**

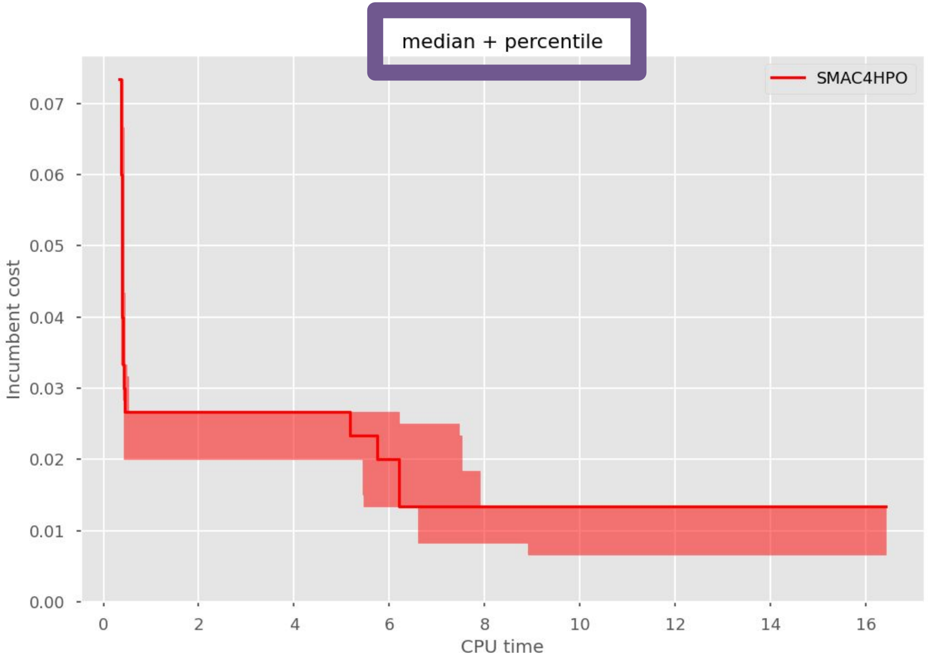


We are interested in the expected performance and the uncertainty on it

- **Expectation** → **Mean**
- **Uncertainty** on expectation → **Standard error** ($\sigma/\sqrt{n}$)

Median and Quantiles

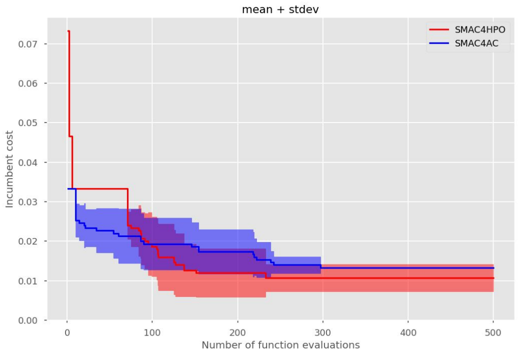
Median across several runs of our HPO tool SMAC over **CPU time**

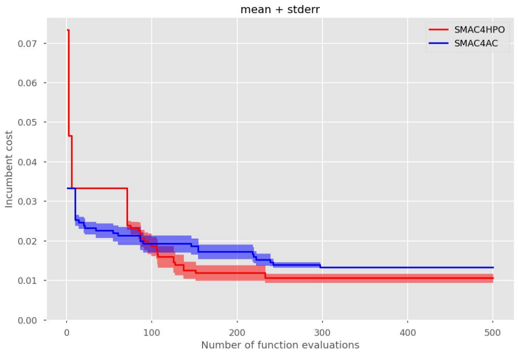


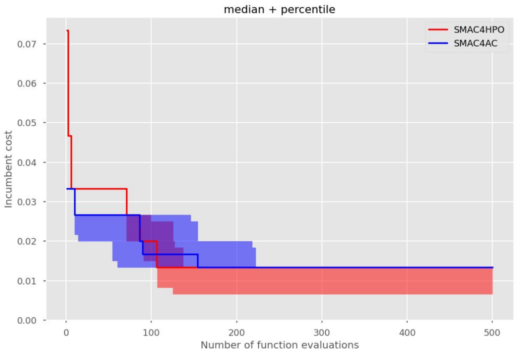
- **Mean** and **stdev/stderr** only make sense if we can assume a **Gaussian** distribution
- **Median** and **Quantiles** are robust and **assumption-free statistics**

Summary Plotting AutoML Performance

1. Plotting anytime performance is important
2. Often better to plot CPU time instead of #evaluations (especially on real benchmarks)
3. Use step functions!
4. Consider log-scales on x and/or y
5. Consider different ways for plotting the uncertainty of cost observations





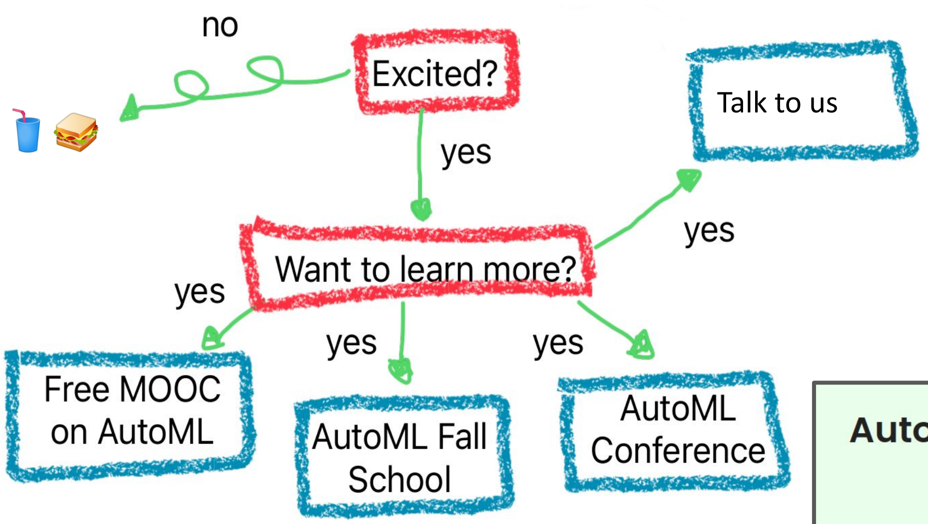


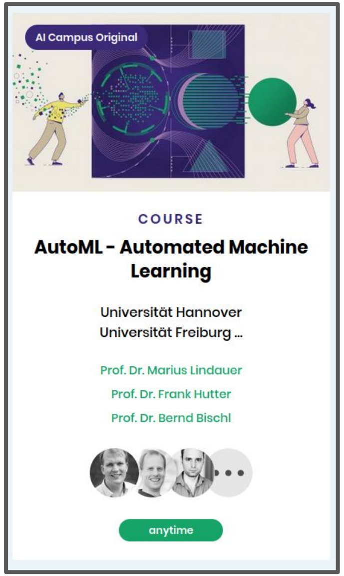
Questions?

Recommendations

- Literature
  - Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges
- Full AutoML
  - Auto-Sklearn
  - AutoGluon
- HPO Tools
  - SMAC3
  - Optuna
  - Syne-Tune
- NAS
  - NNI
  - Auto-PyTorch

Advertisement !!!?!





For more info visit: AutoML.org

Thanks.
See you tomorrow!

👋

References

- A. Canziani, A. Paszke, E. Culurciello (2017) An Analysis of Deep Neural Network Models for Practical Applications (arXiv)
- M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? (JMLR)
- A. Zela, A. Klein, S. Falkner, F. Hutter (2018). Towards Automated Deep Learning: Efficient Joint Neural Architecture and Hyperparameter Search (AutoML@ICML)