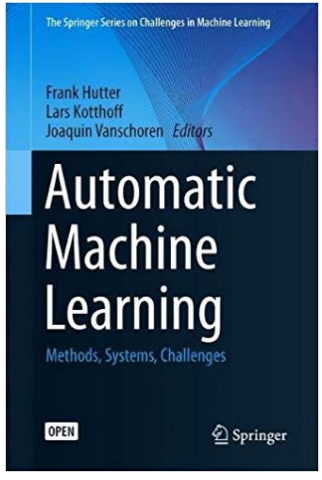
# Automated Machine Learning

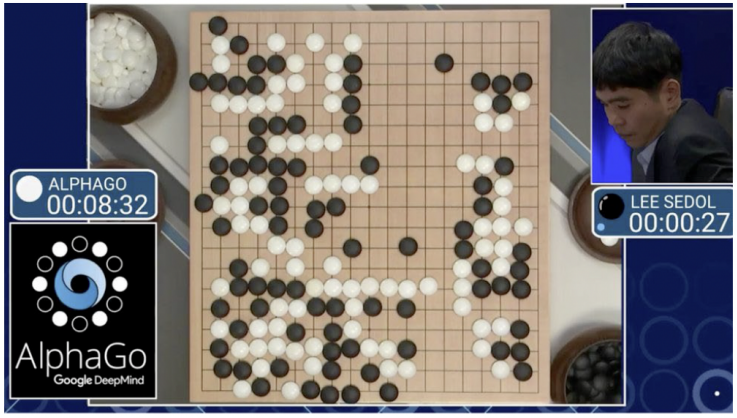
Stéphane Canu
asi.insa-rouen.fr/enseignants/~scanu
scanu@insa-rouen.fr




INSA Rouen Normandie

# Motivation

The importance of Hyperparameter optimization (HPO)




**AlphaGo**: tuning 10 hyperparameters improved win rate from 50% to 65% before playing Lee Sedol

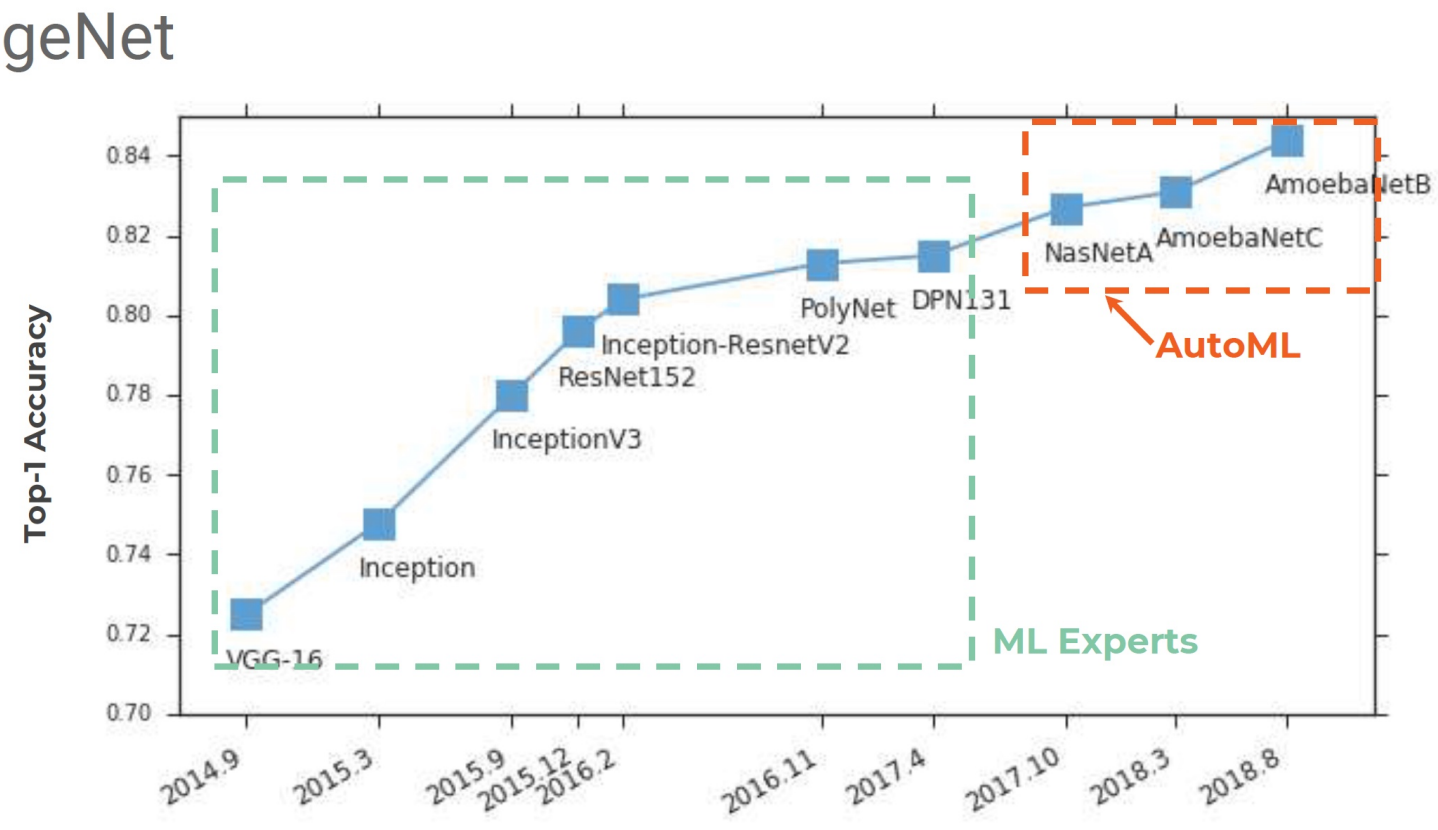
| Hyperparameter Group | Hyperparameters |
|---|---|
| **Finetuning Strategies** | Percentage of the Model to Freeze, Layer Decay, Linear Probing, Stochastic Norm, SP-Regularization, DELTA Regularization, BSS Regularization, Co-Tuning |
| **Regularization Techniques** | MixUp, MixUp Probability*, CutMix, Drop-Out, Label Smoothing, Gradient Clipping |
| **Data Augmentation** | Data Augmentation Type (Trivial Augment, Random Augment, Auto-Augment), Auto-Augment Policy*, Number of operations*, Magnitude* |
| **Optimization** | Optimizer type (SGD, SGD+Momentum, Adam, AdamW, Adamp), Beta-s*, Momentum*, Learning Rate, Warm-up Learning Rate, Weight Decay, Batch Size |
| **Learning Rate Scheduling** | Scheduler Type (Cosine, Step, Multi-Step, Plateau), Patience*, Decay Rate*, Decay Epochs* |

Hyperparameters for fine-tuning foundation models

Too many choices [Pineda et al, 2023]

# Motivation

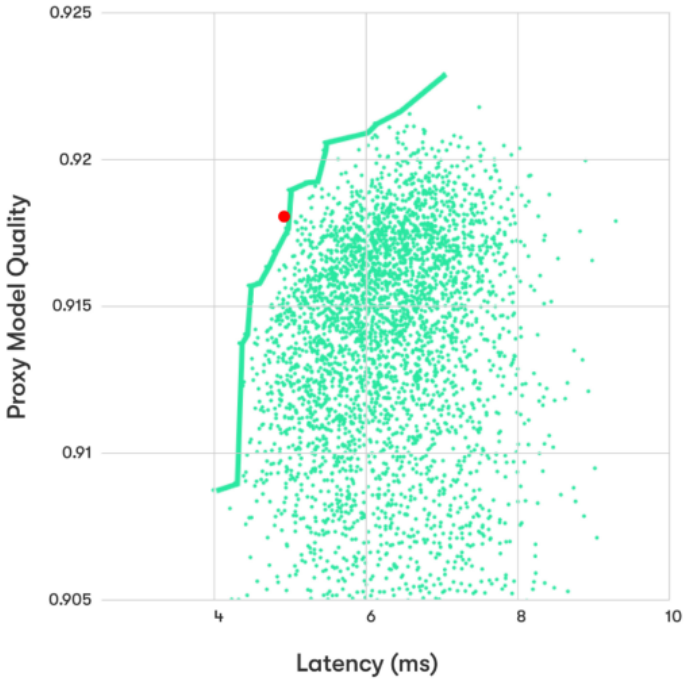
Quoc Le from Google « *AI Frontiers Conference* »nov 2018 « *Using Machine Learning to Automate Machine Learning* »



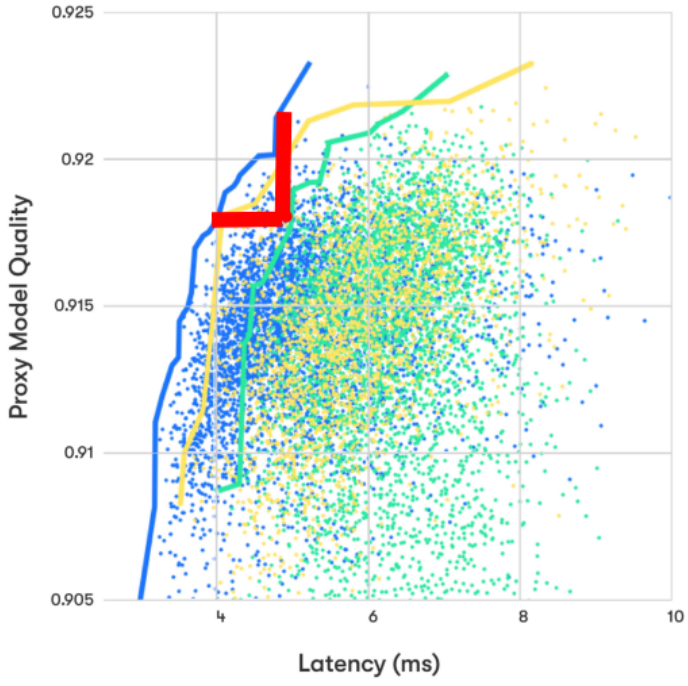

slideshare.net/AIFrontiers/quoc-le-at-ai-frontiers-using-machine-learning-to-automate-machine-learning

# Motivation

How to build Waymo auto pilot (semantic segmentation task)?



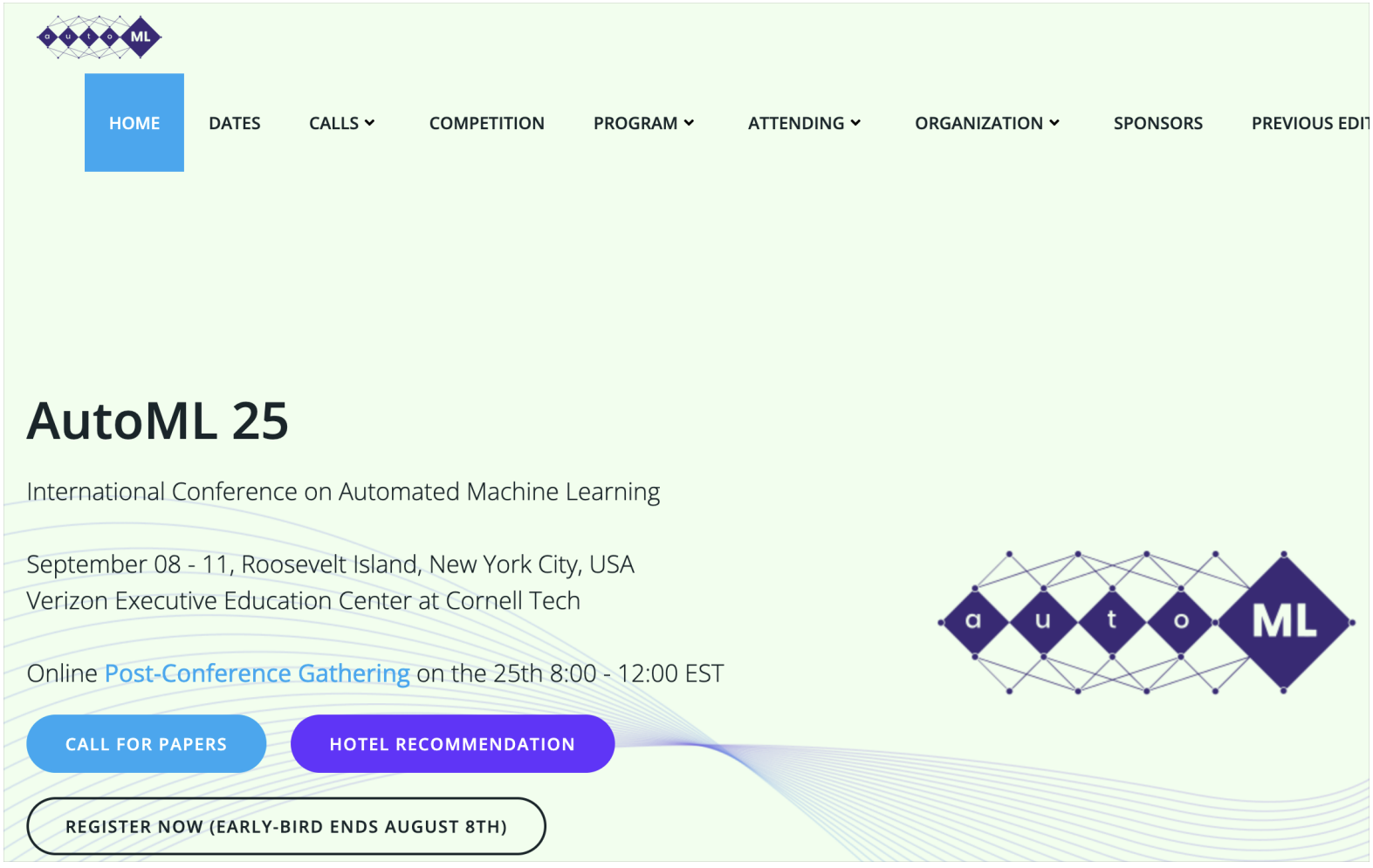

random search




Auto ML .

https://blog.waymo.com/2019/07/automl-automating-design-of-machine.html

# Motivation


HOME
DATES
CALLS
COMPETITION
PROGRAM
ATTENDING
ORGANIZATION
SPONSORS
PREVIOUS EDI
AutoML 25
International Conference on Automated Machine Learning
September 08 - 11, Roosevelt Island, New York City, USA
Verizon Executive Education Center at Cornell Tech
Online Post-Conference Gathering on the 25th 8:00 - 12:00 EST
CALL FOR PAPERS
HOTEL RECOMMENDATION
REGISTER NOW (EARLY-BIRD ENDS AUGUST 8TH)

# Motivation

kaggle

Create

Home

Competitions

Datasets

Models

Code

Discussions

Learn

More

Search

## 2024 AutoML Grand Prix

Enter this multi-competition challenge to test the best Automated Machine Learning Tools!



**Register Here** **Official Competition Rules**

Start your engines for the AutoML Grand Prix! Kaggle is excited to partner with the International Conference on Automated Machine Learning for this multi-competition challenge to test the best AutoML practitioners!

## What is AutoML?

Automated machine learning (AutoML) tools quickly build machine learning models on raw data with minimal input from a developer. AutoMLs not only empower people without machine learning expertise to build and use effective models, but can quickly and easily establish early benchmarks. AutoMLs can make ML methods more efficient, robust trustworthy, and available to everyone.

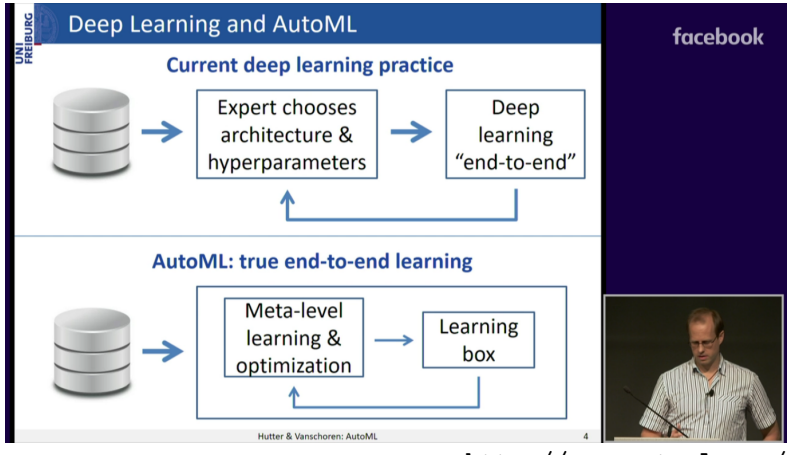
## How the AutoML Grand Prix works

**Overview**

1. Register your team using the Registration Form
    - Please note- you must participate on the same team throughout the competition to be eligible for Grand Prix

# Lecture road map

1. AutoML and the machine learning process
2. Dataset
3. Preprocessing
4. Hyperparameter tuning
5. Post processing – Model agregation
6. Auto ML frameworks

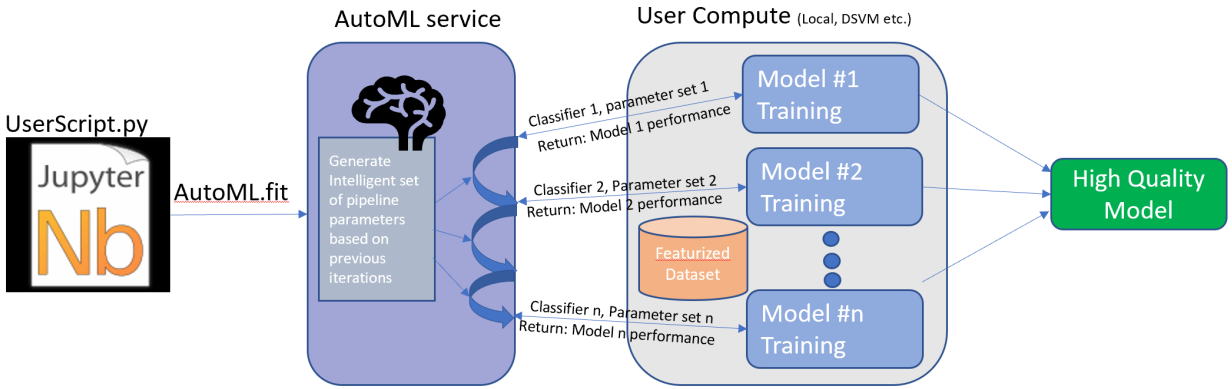



http://www.automl.org/

# AutoML = Learning to learn

- input:
  1. data $\mathcal{S} = (x_i, y_i), i = 1, \ldots, n$
  2. ML algorithm(s) $f_\ell(x, \theta, h)$
     - $x$ input
     - $\theta$ parameters
     - $h$ hyperparameters
  3. other datasets with solutions $(S_j, f_{\ell_j}(x, \theta_j, h_j)), j = 1, \ldots, m$

- output: a decision function $f$ combining $f_{\ell_k}(x, \theta_k, h_k), k = 1, \ldots, K$




Microsoft's automated machine learning

# A simple example

Learning to learn?

- input:
  1. data $\mathcal{S} = (x_i, y_i), i = 1, \ldots, n$ classification y$\in \{0, 1\}$
  2. ML algorithm(s) $f_\ell(x, \theta, h)$

     **SVM** $f_1(x, \theta = (\alpha, b), h = (C, \sigma, \varepsilon))$

     **RF** $f_2(x, \theta, h = (t, p, d, \ldots))$

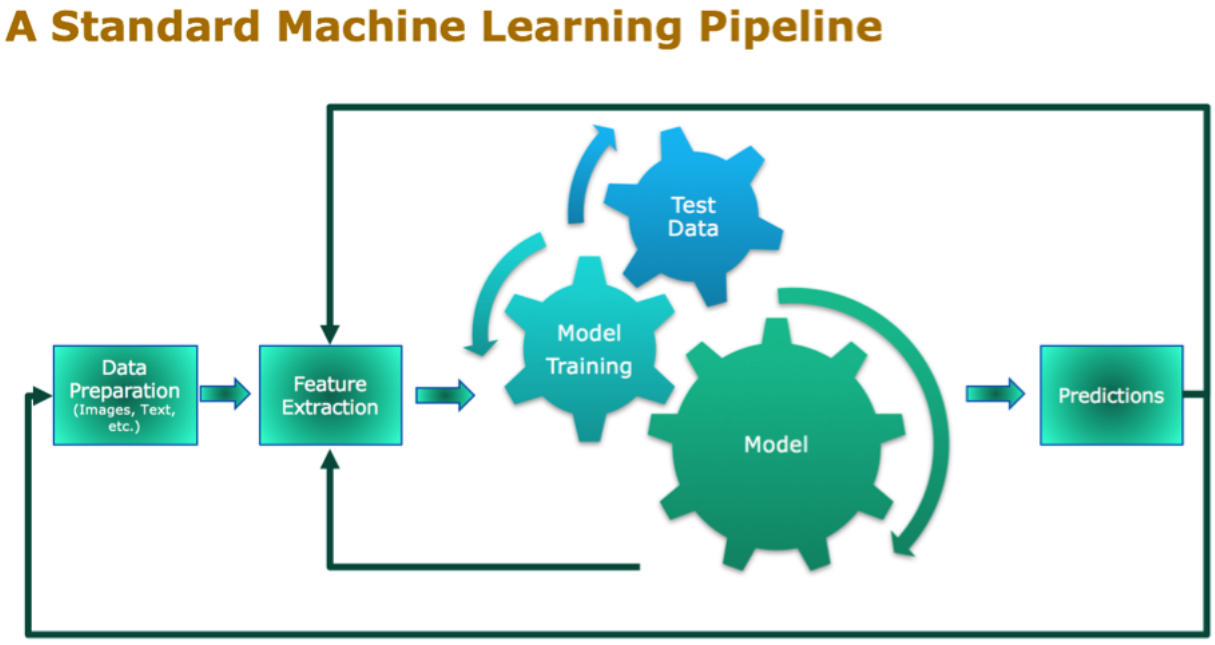
     **kNN** $f_3(x, \theta = \sim, h = k)$

     ...
  3. other datasets with solutions: Iris, glass, cover type,... (UCI)

- output: $K = 3$ models were selected

$$f(x) = \frac{1}{2} f_1(x, (\alpha^\star, b^\star), (1, .1, = 0)) + \frac{1}{3} f_3(x, k = 3) + \frac{1}{6} f_3(x, k = 5)$$

# The ML pipeline




ML algorithm(s) $\rightarrow$ ML Pipeline

- coding/normalization/missing values...
- preprosessing
  - feature generation
  - representation (PCA...)
  - feature selection
  - ...

Easily 20 to 50 design decisions: Time budget issues

# Auto ML research fields

- hyperparameter optimization (HPO)
  GridSearchCV

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svc = svm.SVC(gamma="scale")
clf = GridSearchCV(estimator=svc, param_grid=parameters, cv=5)
```

- NAS = neural architecture search (specific hyper parameter)
- combining models
- meta learning (long term goal)
  sklearn.pipeline

```
from sklearn.pipeline import Pipeline

logistic = SGDClassifier(loss='log', penalty='l2',
early_stopping=True, max_iter=10000, tol=1e-5, random_state=0)
pca = PCA()
pipe = Pipeline(steps=[('pca', pca), ('logistic', logistic)])
```

- https://github.com/skrub-data/skrub

# Problem statement: automated machine learning

Define a search space $\mathcal{A}$

$$\begin{array}{ll} \min\limits_{a \in \mathcal{A}} & \mathcal{L}_{val}(w^\star(a), a) \\ \text{avec} & w^\star(a) = \arg\min\limits_{w} \mathcal{L}_{\text{train}}(w, a) \end{array}$$

## 2 Problem Statement: Automated Machine Learning

Let $P(\mathcal{D})$ be a distribution of datasets from which we can sample an individual dataset's distribution $P_d = P(\mathbf{X}, \mathbf{y})$. The AutoML problem is to generate a trained pipeline $\mathcal{M}_{\boldsymbol{\lambda}} : \mathbf{x} \mapsto y$, hyper-parameterized by $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ that automatically produces predictions for samples from the distribution $P_d$ minimizing the generalization error:

$$GE(\mathcal{M}_{\boldsymbol{\lambda}}) = \int \mathcal{L}(\mathcal{M}_{\boldsymbol{\lambda}}(\mathbf{x}), y) P_d(\mathbf{x}, y) d\mathbf{x} dy. \quad (1)$$

Since a dataset can only be observed through a set of $n$ independent observations $\mathcal{D}_d = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \sim P_d$, we can only empirically approximate the generalization error on sample data:

$$\widehat{GE}(\mathcal{M}_{\boldsymbol{\lambda}}, \mathcal{D}_d) = \frac{1}{|\mathcal{D}_d|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_d} \mathcal{L}(\mathcal{M}_{\boldsymbol{\lambda}}(\mathbf{x}_i), y_i). \quad (2)$$

AutoML systems automatically search for the best $\mathcal{M}_{\boldsymbol{\lambda}^*}$:

$$\mathcal{M}_{\boldsymbol{\lambda}^*} \in \operatorname*{argmin}_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \widehat{GE}(\mathcal{M}_{\boldsymbol{\lambda}}, \mathcal{D}_{\text{train}}) \quad (3)$$

and estimate GE, e.g., by a $k$-fold cross validation:

$$\widehat{GE}_{\text{CV}}(\mathcal{M}_{\boldsymbol{\lambda}}, \mathcal{D}_{\text{train}}) = \frac{1}{k} \sum_{i=1}^{k} \widehat{GE}(\mathcal{M}_{\boldsymbol{\lambda}}^{\mathcal{D}_{\text{train}}^{(\text{train},i)}}, \mathcal{D}_{\text{train}}^{(\text{val},i)}) \quad (4)$$

where $\mathcal{M}_{\boldsymbol{\lambda}}^{\mathcal{D}_{\text{train}}^{(\text{train},i)}}$ denotes that $\mathcal{M}_{\boldsymbol{\lambda}}$ was trained on the $i$-th training fold $\mathcal{D}_{\text{train}}^{(\text{train},i)}$. Assuming that an AutoML system can select via $\boldsymbol{\lambda}$ both, the algorithm and its hyperparameter settings, this definition using $\widehat{GE}_{\text{CV}}$ is equivalent to the definition of the CASH problem [2], [4].

### 2.1 Time-bounded AutoML

In practice, users are not only interested to obtain an optimal pipeline $\mathcal{M}_{\boldsymbol{\lambda}^*}$ eventually, but have constraints on how much time and compute resources they are willing to invest. We denote the time it takes to evaluate $\widehat{GE}(\boldsymbol{\lambda}, \mathcal{D}_{\text{train}})$ as $t_{\boldsymbol{\lambda}}$ and the overall optimization budget by $T$. Our goal is to find

$$\mathcal{M}_{\boldsymbol{\lambda}^*} \in \operatorname*{argmin}_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \widehat{GE}(\boldsymbol{\lambda}, \mathcal{D}_{\text{train}}) \text{ s.t. } \left(\sum t_{\boldsymbol{\lambda}}\right) < T \quad (5)$$

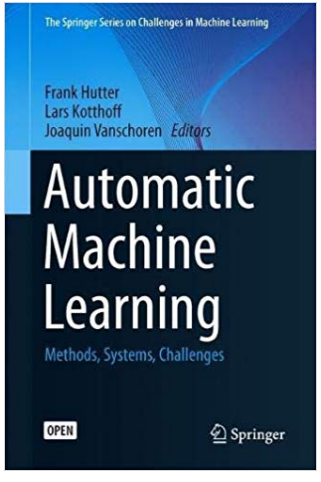
where the sum is over all pipelines evaluated, explicitly honouring the optimization budget $T$.

### 2.2 Generalization of AutoML

Ultimately, a well performing and robust optimization policy $\pi : \mathcal{D} \mapsto \mathcal{M}_{\boldsymbol{\lambda}}^{\mathcal{D}}$ of an AutoML system should not only perform well on a single dataset but on the entire distribution over datasets $P(\mathcal{D})$. Therefore, the meta-problem of

# Lecture road map

1. AutoML and the machine learning process
2. Dataset
3. Preprocessing
4. Hyperparameter tuning
5. Post processing – Model agregation
6. Auto ML frameworks



# Real-world benchmark datasets for ML

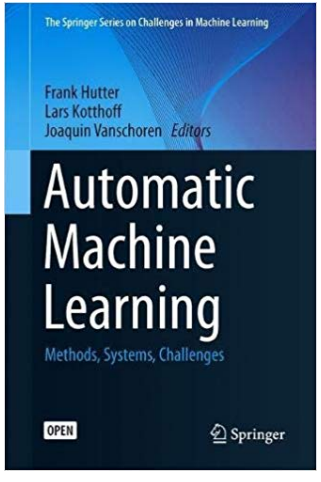
- UCI Machine Learning Archive: 488 Data Sets
  `https://archive.ics.uci.edu/ml/datasets.php`



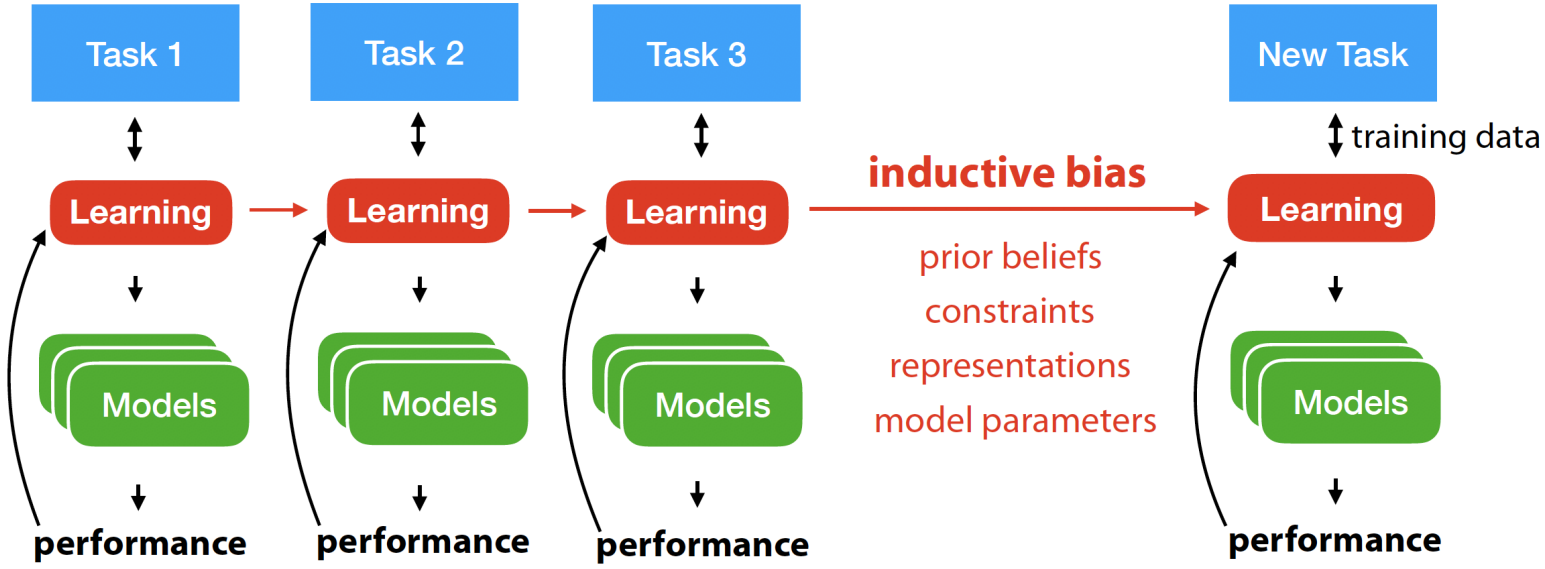

- OpenML: 2972 results
  `https://www.openml.org/search?type=data`
- Kaggle: 24171 dataset
  `https://www.kaggle.com/datasets`
- Wikipadia List of datasets for machine-learning research
  `https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research`
- PMLB: A large, curated repository of benchmark datasets for evaluating supervised machine learning algorithms
  `https://github.com/EpistasisLab/penn-ml-benchmarks`

# Lecture road map

1. AutoML and the machine learning process
2. Dataset
3. Preprocessing
4. Hyperparameter tuning
5. Post processing – Model agregation
6. Auto ML frameworks



# Preprocessing: Meta-learning




## Toward meta-data: use 144 UCI data sets

- Represent the problems: $X$
- Find similar problems: $d(new, X)$
- Adapt the configuration: algorithm, pipeline, hyper-parameters...

# How to represent a problem?

Use meta-feature

- Data $D$:
  - Number of instances, features, classes, missing values, outliers...
  - Statistical: skewness, kurtosis, corr., sparsity, entropy, mutual info...
- Model-based $T$: properties of simple a decision tree trained the data
- Landmarkers $P$: performance of fast algorithms trained on the task

$X = (D, T, P)$

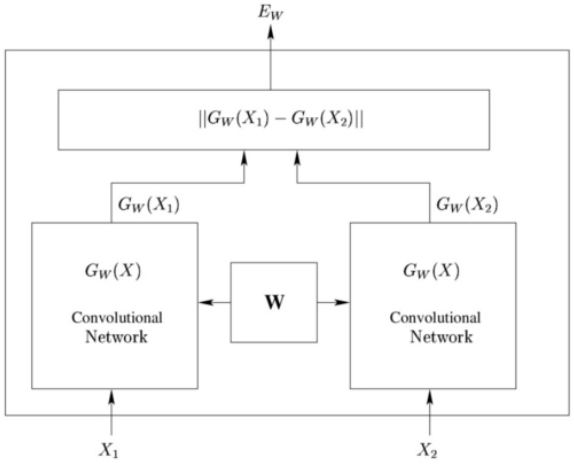
| | | | |
|---|---|---|---|
| Equiv. nr. feats | $\frac{H(C)}{MI(C,X)}$ | Intrinsic dimensionality (Michie et al., 1994) | |
| Noise-signal ratio | $\frac{H(X)-MI(C,X)}{MI(C,X)}$ | Noisiness of data (Michie et al., 1994) | |
| Fisher's discrimin. | $\frac{(\mu_{c1}-\mu_{c2})^2}{\sigma_{c1}^2-\sigma_{c2}^2}$ | Separability classes $c_1, c_2$ (Ho and Basu, 2002) | See Ho:2002 |
| Volume of overlap | | Class distribution overlap (Ho and Basu, 2002) | See Ho and Basu (2002) |
| Concept variation | | Task complexity (Vilalta and Drissi, 2002) | See Vilalta (1999) |
| Data consistency | | Data quality (Köpf and Iglezakis, 2002) | See Köpf and Iglezakis (2002) |
| Nr nodes, leaves | $\lvert\eta\rvert, \lvert\psi\rvert$ | Concept complexity (Peng et al., 2002) | Tree depth |
| Branch length | | Concept complexity (Peng et al., 2002) | min,max,$\mu$,$\sigma$ |
| Nodes per feature | $\lvert\eta_X\rvert$ | Feature importance (Peng et al., 2002) | min,max,$\mu$,$\sigma$ |
| Leaves per class | $\frac{\lvert\psi_c\rvert}{\lvert\psi\rvert}$ | Class complexity (Filchenkov and Pendryak, 2015) | min,max,$\mu$,$\sigma$ |
| Leaves agreement | $\frac{n_{\psi_j}}{n}$ | Class separability (Bensusan et al., 2000) | min,max,$\mu$,$\sigma$ |
| Information gain | | Feature importance (Bensusan et al., 2000) | min,max,$\mu$,$\sigma$, gini |
| Landmarker(1NN) | $P(\theta_{1NN}, t_j)$ | Data sparsity (Pfahringer et al., 2000) | See Pfahringer et al. (2000) |
| Landmarker(Tree) | $P(\theta_{Tree}, t_j)$ | Data separability (Pfahringer et al., 2000) | Stump,RandomTree |
| Landmarker(Lin) | $P(\theta_{Lin}, t_j)$ | Linear separability (Pfahringer et al., 2000) | Lin.Disciminant |
| Landmarker(NB) | $P(\theta_{NB}, t_j)$ | Feature independence (Pfahringer et al., 2000) | See Ler et al. (2005) |

# Find similar problems

Compute a distance between problems:

- k-NN
- Deep metric learning: e.g. Siamese Network
  Output (Target): A label, 0 for similar, 1else.
  - Similar: loss $= \min_W \|G_W(X_1) - G_W(X_2)\|^2$
  - Else: loss $=$

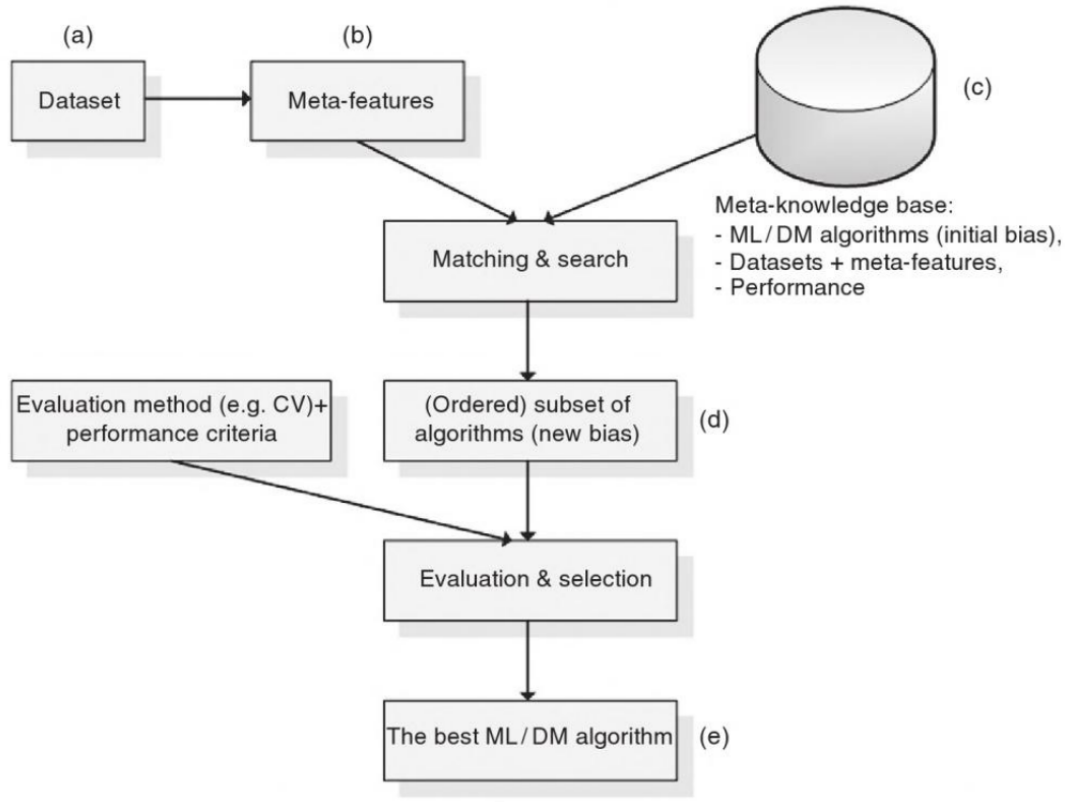
$$\min_W max(m - \|G_W(X_1) - G_W(X_2)\|, 0)^2$$




NeurIPS 1994 Bromley et al. "Signature verification using a" siamese" time delay neural network"

# Recommend and adapt configurations

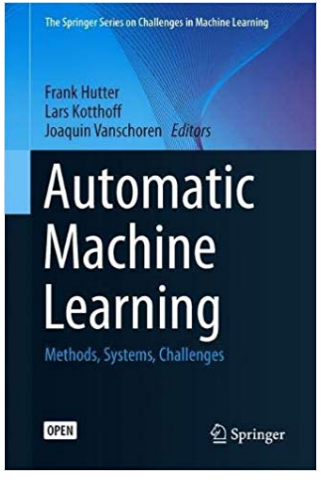
- Zero-shot meta-models: take the best
- Ranking weighted models
- Randomly pick, tune and adapt




Brazdil et al. "Metalearning: Applications to Data Mining", 2008

# Lecture road map

1. AutoML and the machine learning process
2. Dataset
3. Preprocessing
4. Hyperparameter tuning
5. Post processing – Model agregation
6. Auto ML frameworks



# Hyperparameter tunning

Different types of hyperparameters

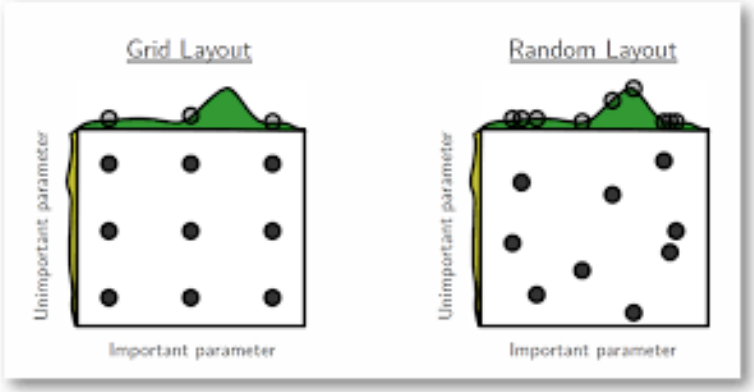
- continuous (SVM's $C$, learning rate)
- integer (kNN's $k$, number of layer)
- categorical
  - algorithm choice
  - activation function
  - ...

Easily 20 to 50 design decisions: Time budget issues

# Hyperparameter tunning

Use preprocessing to find out the domain

- grid search
- random search
- reinforcement learning
- evolutionary strategy
- Hyperparameter Gradient Descent [Franceschi et al, ICML 2018]
- Bayesian search

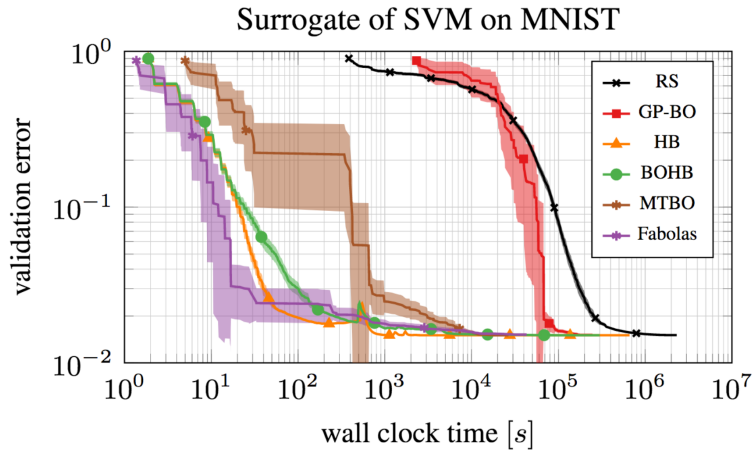



J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization.", JMLR 2012

# how to manage the time budget to scale

- Brute force (time out)

- Many cheap evaluations on small data
  Few expensive evaluations on all data




- Successive Halving (SH) [Jamieson & Talwalkar, AISTATS 2016]
  - begin with traininng $m$ models
  - at each time step keep half of them

- Hyperband, [Li et al, ICLR 2017] a bandit strategy that dynamically allocates resources to a set of random configurations and uses successive halving

- Bayesian optim. [Falkner, Kleil, ICML 2018]

Falkner et al.. "BOHB: Robust and Efficient Hyperparameter Optimization at Scale", JMLR 2018

# Hyperparameter auto tuning tools

| Package | Complex Hyperparameter Spaces | Multi-Objective | Multi-Fidelity | Instances | CLI | Parallelism |
|---|---|---|---|---|---|---|
| HyperMapper | ✅ | ✅ | ❌ | ❌ | ❌ | ❌ |
| Optuna | ✅ | ✅ | ✅ | ❌ | ✅ | ✅ |
| Hyperopt | ✅ | ❌ | ❌ | ❌ | ✅ | ✅ |
| BoTorch | ❌ | ✅ | ✅ | ❌ | ❌ | ✅ |
| OpenBox | ✅ | ✅ | ❌ | ❌ | ❌ | ✅ |
| HpBandSter | ✅ | ❌ | ✅ | ❌ | ❌ | ✅ |
| SMAC | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ |

last update of table in 2021

`https://github.com/automl/HpBandSter`
`https://github.com/automl/SMAC3`

by M. Lindauer and F. Hutter, Lecture at AutoML Fall School 2022: The AutoML Landscape

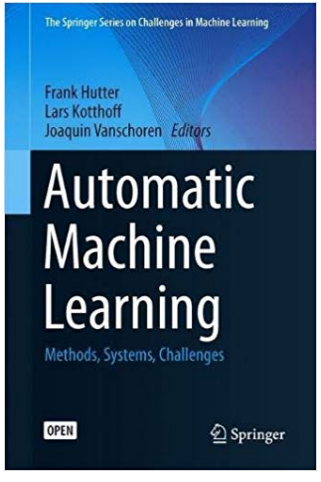
# Results on the neural architecture search




Regularized Evolution for Image Classifier Architecture Search E. Real et al, 2018

# Lecture road map

1. AutoML and the machine learning process
2. Dataset
3. Preprocessing
4. Hyperparameter tuning
5. Post processing – Model agregation
6. Auto ML frameworks



# Post processing: Model agregation (Stacking)

## Model construction

- Predictor generation: $f_1(X), \ldots, f_K(X)$
- Feature: $\Phi(X) = (f_1(X), \ldots, f_K(X))^T$...

## Penalized Loss

- Minimization of

$$\arg\min_{\beta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, < \Phi(X_i), \beta >) + pen(\beta)$$

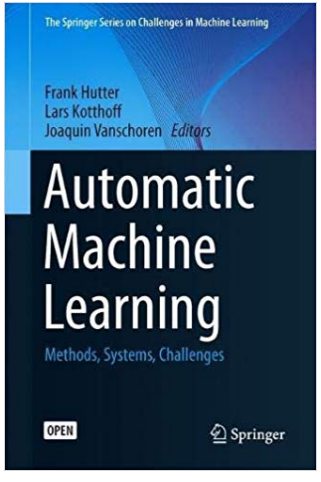
  where $pen(\theta)$ is a penalty.
- Predictor selection if $\beta$ is sparse.

## Classical Penalties

- AIC / Ridge / Lasso / Elastic Net

# Lecture road map

1. AutoML and the machine learning process
2. Dataset
3. Preprocessing
4. Hyperparameter tuning
5. Post processing – Model agregation
6. Auto ML frameworks



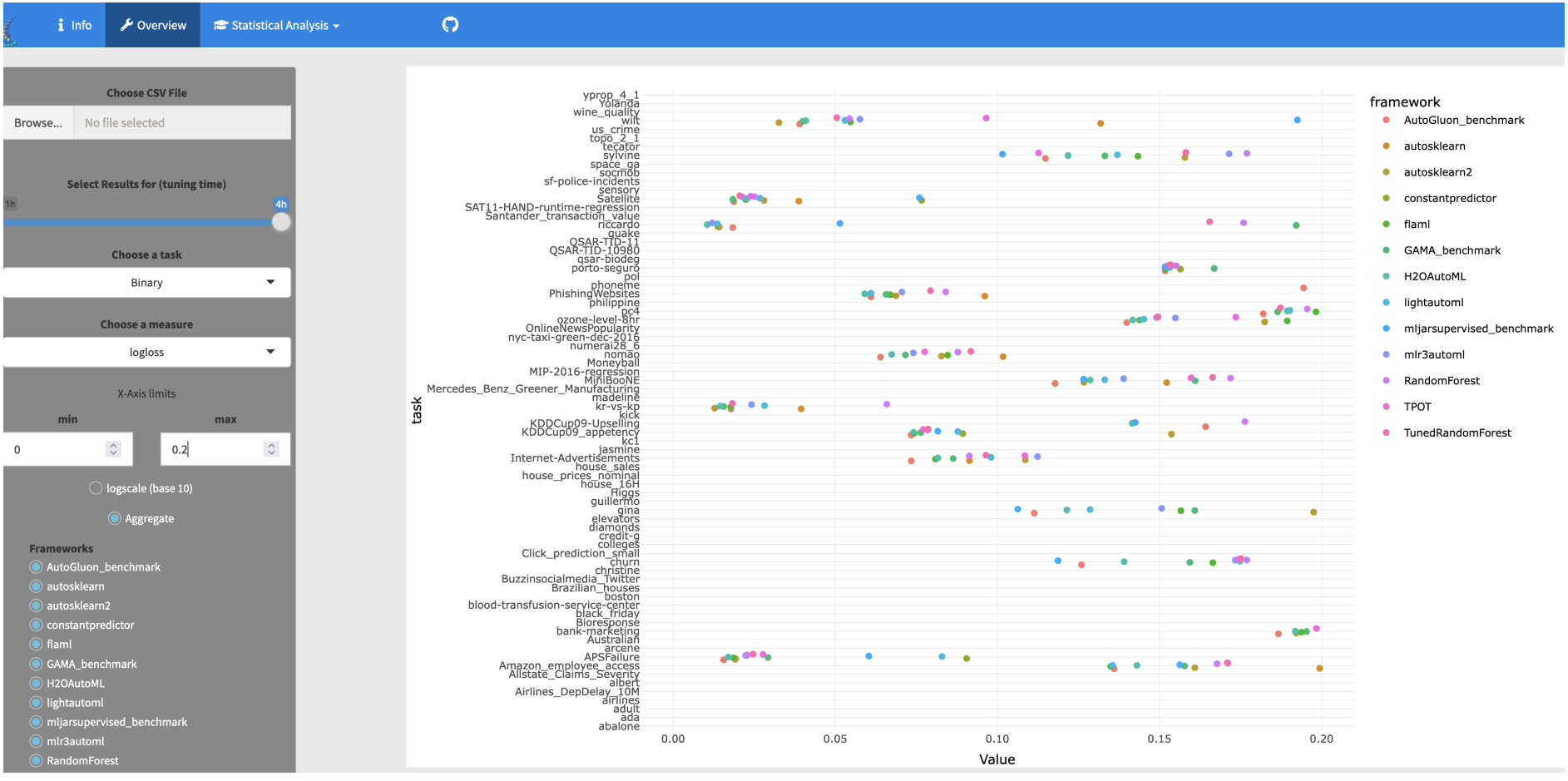
# Auto ML frameworks

- based on frameworks
  - H2O `docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html` build on H2O framework
  - Auto Sklearn `automl.github.io/auto-sklearn/master/` build on SKLearn framework
  - Auto-WEKA `www.automl.org/automl/autoweka/` build on WEKA framework
- commercial:
  - Google (Vizier, automlvision, automl NLP), Azure, FBLearner...
- research (other)
  - TPOT `github.com/EpistasisLab/tpot` Tree-Based Pipeline Opti.
  - MLJar `github.com/mljar/mljar-supervised`
  - AutoGluon (Amazon) `github.com/awslabs/autogluon`
  - http://news.mit.edu/2019/nonprogrammers-data-science-0115

  ...
- AutoML Frameworks

  `openml.github.io/automlbenchmark/frameworks.html`

# Auto ML Benchmark




https://compstat-lmu.shinyapps.io/AutoML-Benchmark-Analysis/

# Auto ML tools

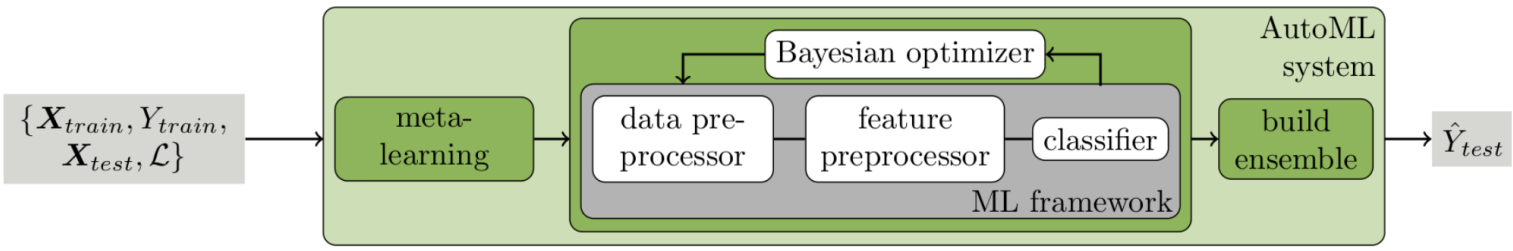



- Auto WEKA [Thornton et al, KDD 2013]
  - Based on WEKA and SMAC Hyperopt sklearn [Komer et al, 2014]
- Auto sklearn [Feurer al, NIPS 2015]
  - Based on scikit learn & SMAC / BOHB
  - Won AutoML competitions 2015-2016 & 2017-2018
- Auto pytorch [Zimmer et al., IEEE PAMI 2021]
  - optimizes the network architecture (NAS) and the hyperparameters
- TPOT [Olson et al, EvoApplications 2016]
  - Based on scikit learn and evolutionary algorithms
- H2O AutoML
  - Based on random search and stacking

`https://www.automl.org/`

# Auto ML at work: Auto Sklearn

## auto-sklearn in one image




## auto-sklearn in four lines of code

```
import autosklearn.classification
cls = autosklearn.classification.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

# Auto ML at work: Auto Sklearn

## 6. AutoML

https://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning
https://automl.github.io/auto-sklearn/stable/

and for the doc https://automl.github.io/auto-sklearn/stable/api.html

In [19]:
```
import warnings
warnings.filterwarnings("ignore")

import autosklearn.classification
import sklearn.model_selection
import sklearn.datasets
import sklearn.metrics

from autosklearn.metrics import make_scorer

automl = autosklearn.classification.AutoSklearnClassifier(
     time_left_for_this_task=7200,
         per_run_time_limit=600,
     exclude_estimators=None,
#     include_preprocessors=["no_preprocessing", ],
     exclude_preprocessors=None)

scorer = autosklearn.metrics.make_scorer(
        'f1_score',
        sklearn.metrics.f1_score,
        pos_label=1,
    )

automl.fit(X_train, y_train, metric=scorer)

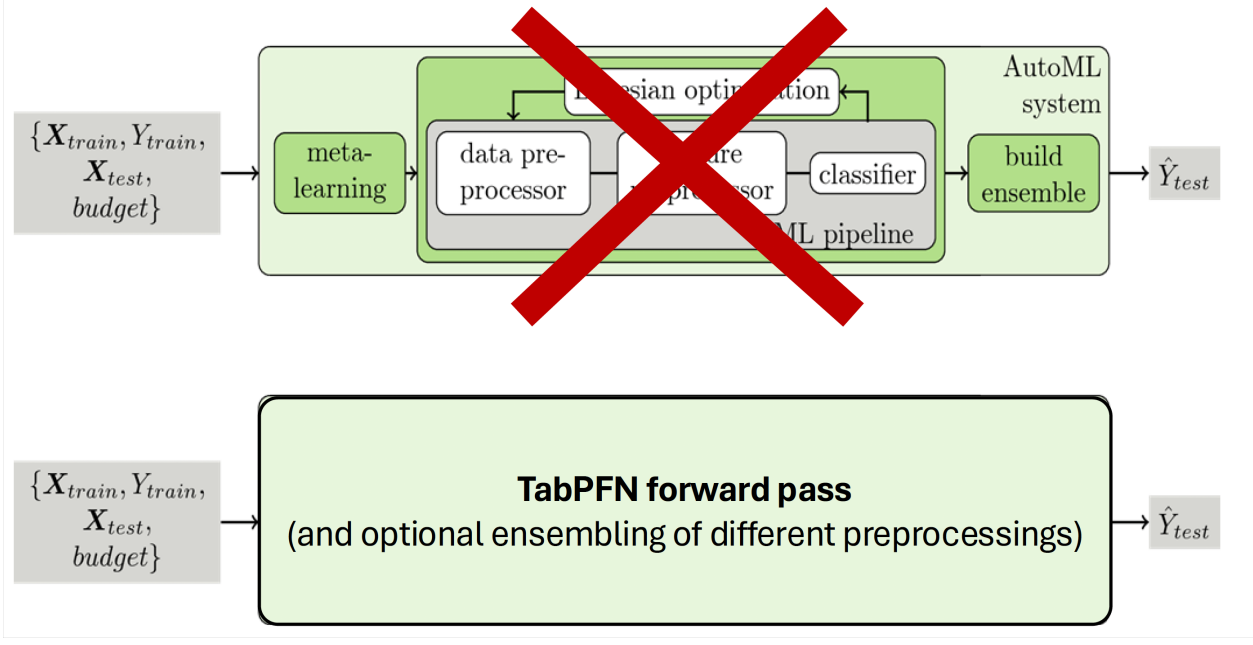
y_pred = automl.predict(X_test)
cf_auto = confusion_matrix(y_test, y_pred)*100/len(y_test)
```

```
[WARNING] [2019-03-18 23:05:40,546:AutoMLSMBO(1)::afcbe3320b92b2a2a9cf1cd0de64b65c] Could not find meta-data director
y /Users/stephane/.local/lib/python3.6/site-packages/autosklearn/metalearning/files/f1_score_binary.classification_de
nse
[WARNING] [2019-03-18 23:05:40,589:EnsembleBuilder(1):afcbe3320b92b2a2a9cf1cd0de64b65c] No models better than random
- using Dummy Score!
[WARNING] [2019-03-18 23:05:40,667:EnsembleBuilder(1):afcbe3320b92b2a2a9cf1cd0de64b65c] No models better than random
```

In [21]:
```
(automl.get_models_with_weights())
```

Out[21]:
```
[(0.2,
  SimpleClassificationPipeline({'balancing:strategy': 'none', 'categorical_encoding:__choice__': 'one_hot_encoding',
 'classifier:__choice__': 'random_forest', 'imputation:strategy': 'mean', 'preprocessor:__choice__': 'no_preprocessing
', 'rescaling:__choice__': 'standardize', 'categorical_encoding:one_hot_encoding:use_minimum_fraction': 'True', 'clas
sifier:random_forest:bootstrap': 'True', 'classifier:random_forest:criterion': 'gini', 'classifier:random_forest:max_
depth': 'None', 'classifier:random_forest:max_features': 0.5, 'classifier:random_forest:max_leaf_nodes': 'None', 'cla
ssifier:random_forest:min_impurity_decrease': 0.0, 'classifier:random_forest:min_samples_leaf': 1, 'classifier:random
_forest:min_samples_split': 2, 'classifier:random_forest:min_weight_fraction_leaf': 0.0, 'classifier:random_forest:n_
estimators': 100, 'categorical_encoding:one_hot_encoding:minimum_fraction': 0.01},
    dataset_properties={
      'task': 1,
      'sparse': False,
      'multilabel': False,
      'multiclass': False,
      'target_type': 'classification',
      'signed': False})),
 (0.18,
  SimpleClassificationPipeline({'balancing:strategy': 'none', 'categorical_encoding:__choice__': 'one_hot_encoding',
 'classifier:__choice__': 'random_forest', 'imputation:strategy': 'mean', 'preprocessor:__choice__': 'no_preprocessing
```

# Auto ML with foundation models

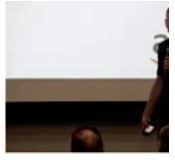



Frank Hutter, Highly Accurate Predictions on Small Data with the Tabular Foundation Model TabPFN

# Conclusion

- How to explore?
- How to evaluate?
- How to combine?

## Automation



- **Model**
  Selection, tuning, validation, monitoring, pretraining
- **Data**
  Discovery, typing, preprocessing, annotation
- **Systems**
  Training, deployment, inference, visualization, automation
- **Users**
  Collaboration, iteration, customization