AUTOMATING MACHINE LEARNING

dr. ir. Joaquin Vanschoren, TU/e

www.openml.org @open_ml, @joavanschoren

AUTOMATING MACHINE LEARNING

Machine learning is often a (black) art

- How much, which data do I need?
- How should I clean the data?
- How should I represent the data (features)?
- How should I preprocess the data?
- Which algorithms should I use?
- How should I tune all algorithm parameters?
- How do I properly evaluate the models?

We can use machine learning to fix this

- Collect data: how algorithms perform on many tasks (OpenML)
- Learn how to learn, build ever better algorithms, models
- Replace trial-and-error by principled automatic processes

AUTOMATIC MACHINE LEARNING

Can algorithms be **trained** to **automatically** build **end-to-end** machine learning systems?

Use machine learning to do better machine learning

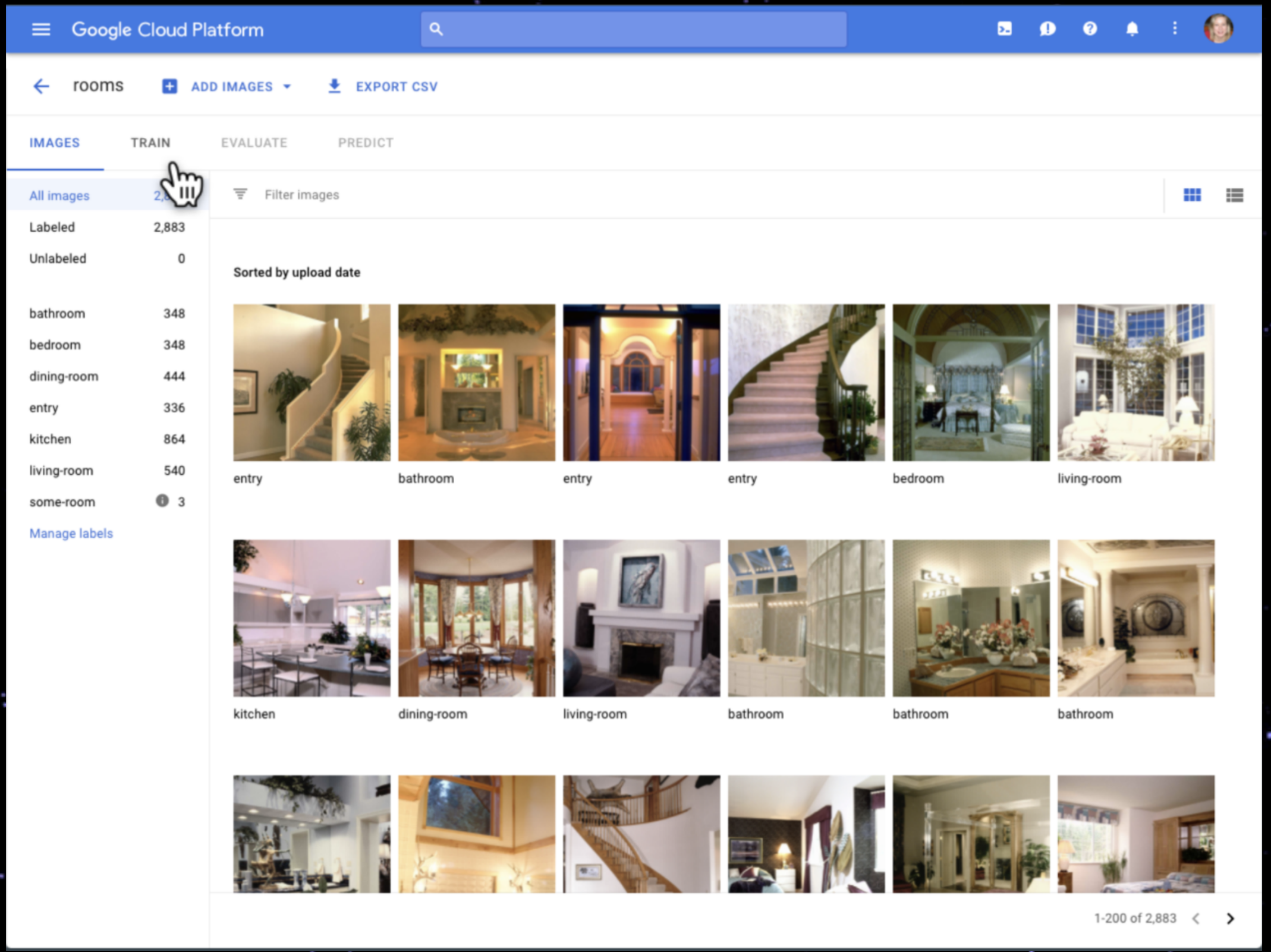
Can we turn

Solution = data + manual exploration + computation

Into

Solution = data + computation (x100)

EXPLOSION OF COMMERCIAL INTEREST



EXPLOSION OF COMMERCIAL INTEREST

H2O.ai
0.0.1

TRAINING DATA

DATASET
creditcard.csv

ROWS	COLUMNS	DROPPED	IGNORED
24K	25	--	--

TARGET COLUMN
default payment next month

TYPE	COUNT	MEAN	STD DEV
Int32	23999	0.2237	0.4167

STATUS: COMPLETE

INTERPRET THIS MODEL

EXPERIMENT SETTINGS

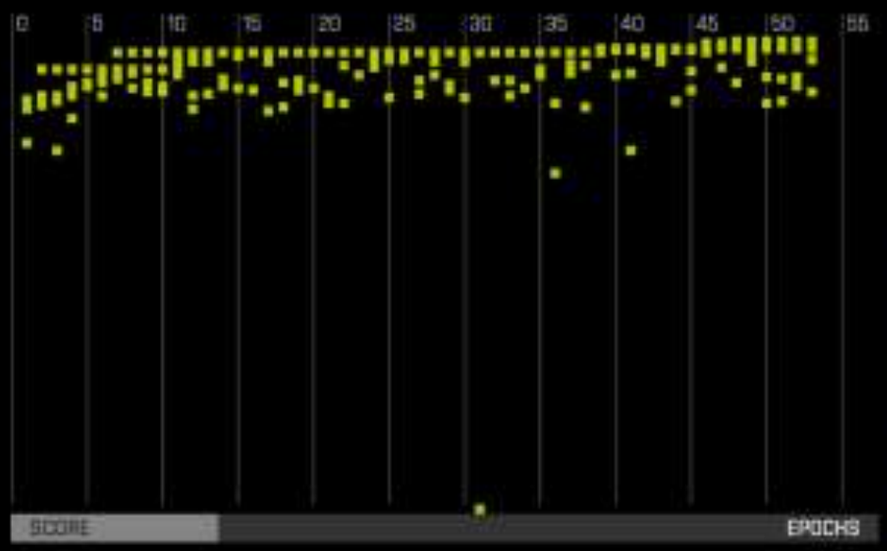
1	53	3	4
WORKERS	ITERATIONS	CV FOLDS	POPULATION

CLASSIFICATION | DETECT IDS | DROP DUPS

GPU STATS

1
2

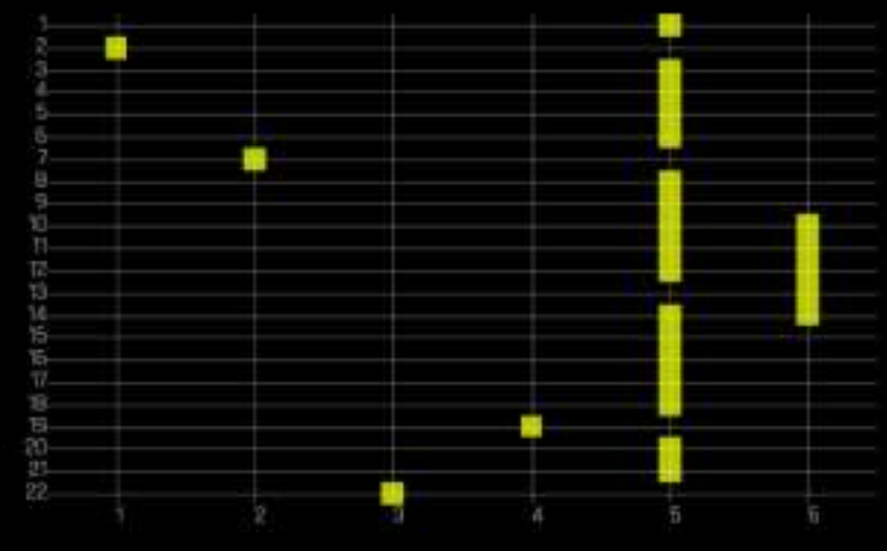
ITERATION SCORES



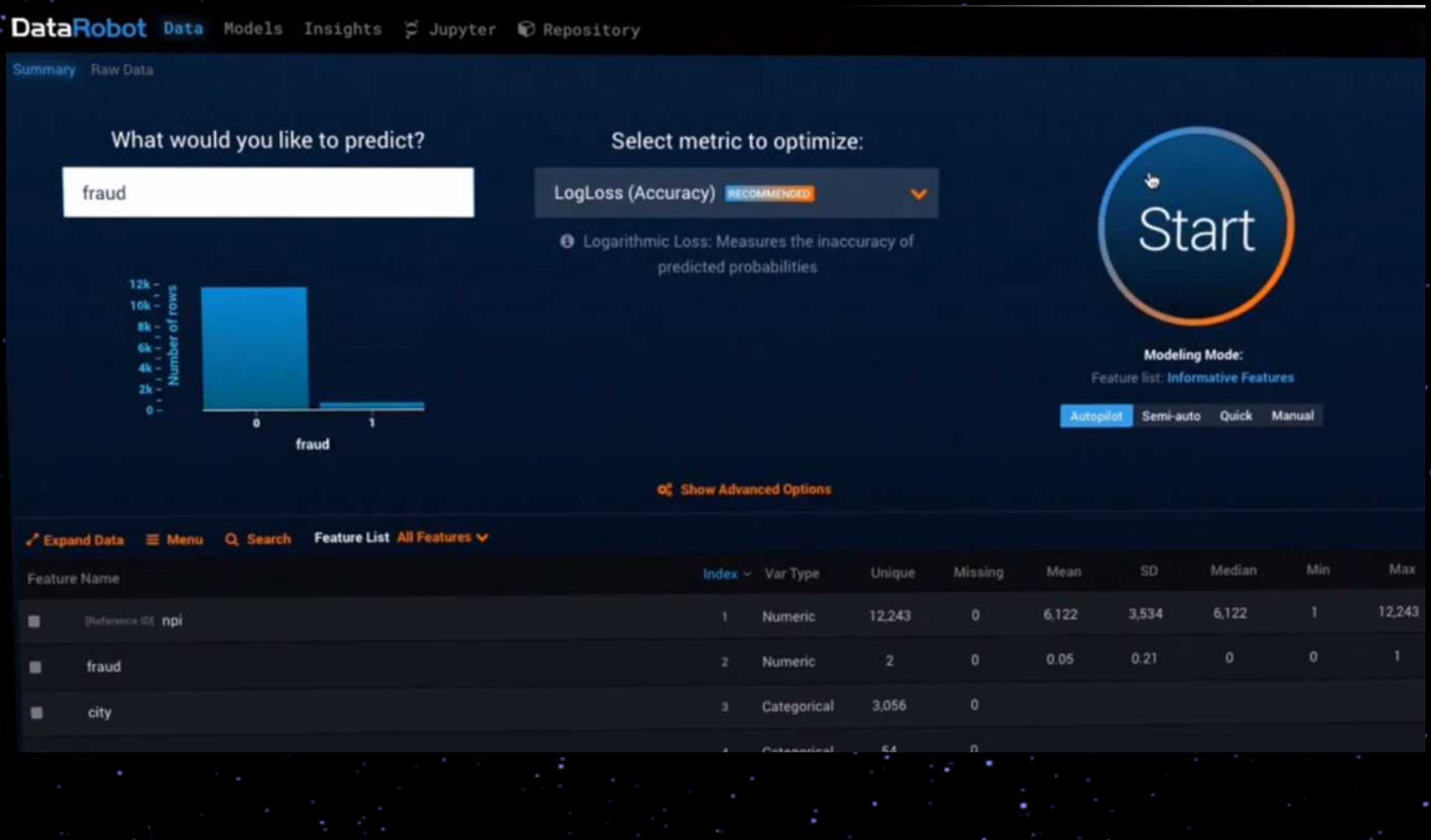
VARIABLE IMPORTANCE

1_PAY_0	7497.51
2_PAY_2	1020.19
5_BILL_AMT1	888.59
0_LIMIT_BAL	825.45
11_PAY_AMT2	777.38
10_PAY_AMT1	584.03
20_CV_TE_SEX_EDUCATION_0	534.89
12_PAY_AMT3	503.78
13_PAY_AMT4	499.84
6_BILL_AMT2	428.59
9_BILL_AMT6	425.90
14_PAY_AMT5	418.09
15_PAY_AMT6	409.82
3_PAY_4	326.66

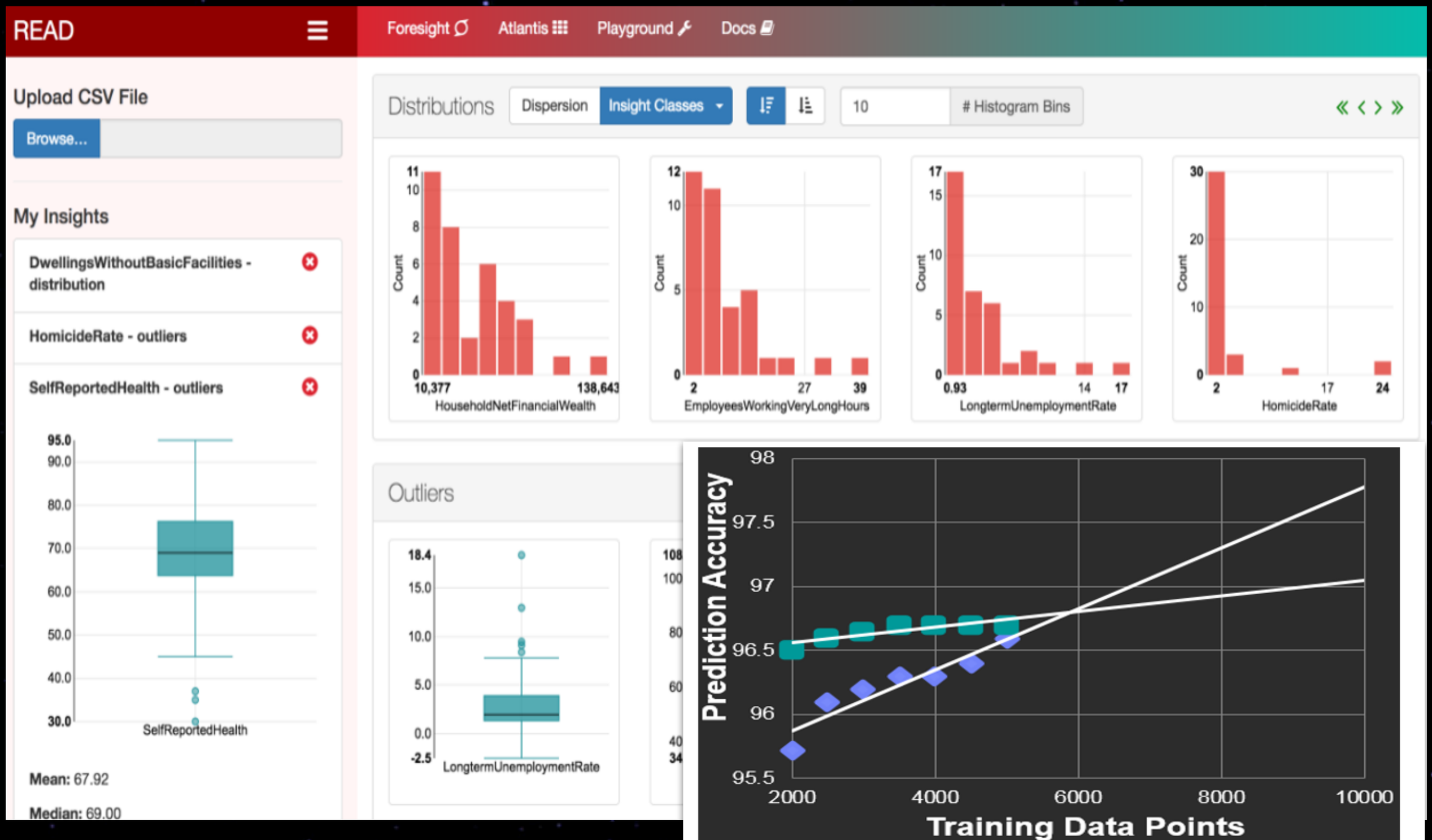
FEATURE TRANSFORMATIONS



EXPLOSION OF COMMERCIAL INTEREST

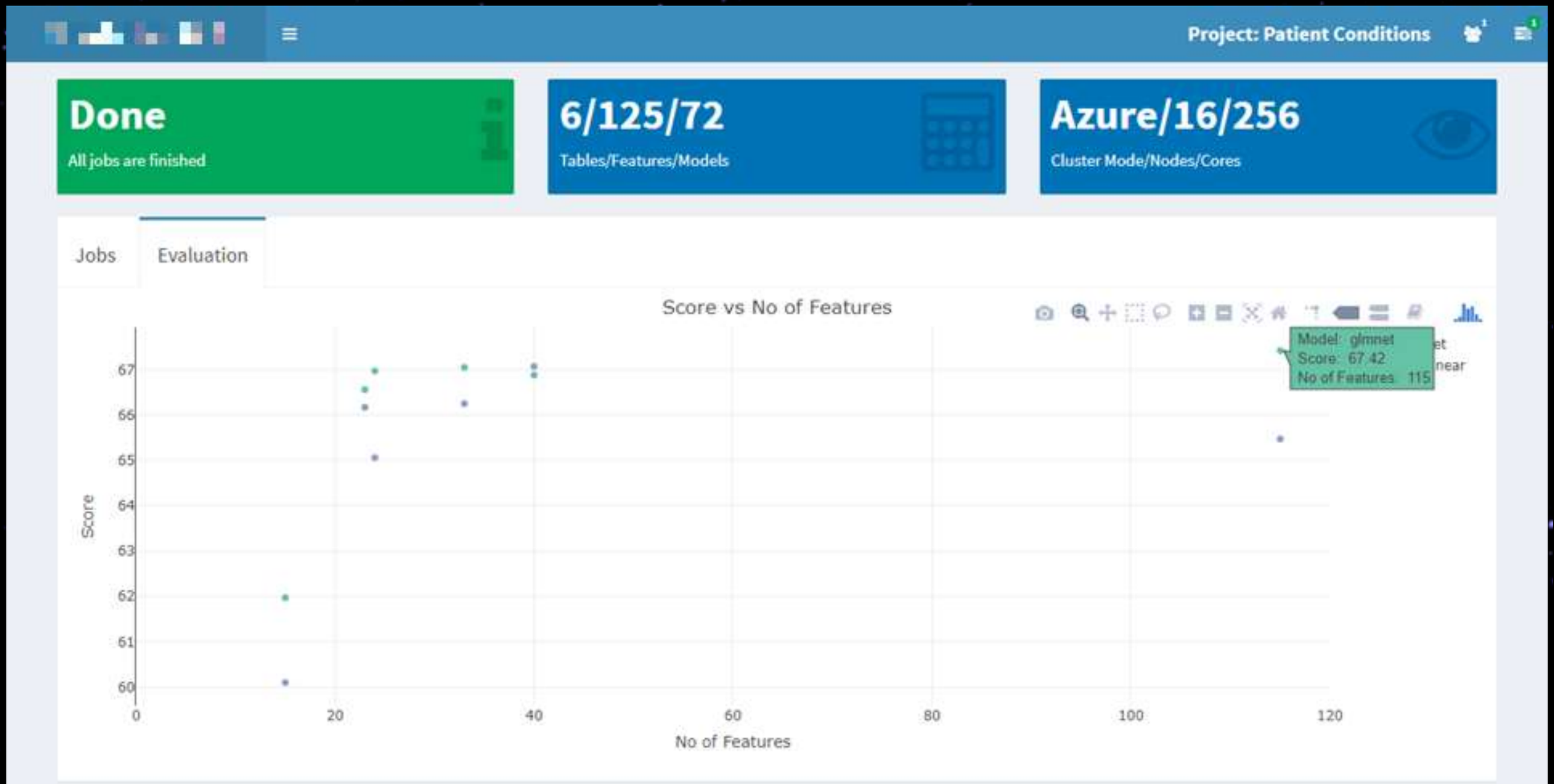


EXPLOSION OF COMMERCIAL INTEREST



(IBM)

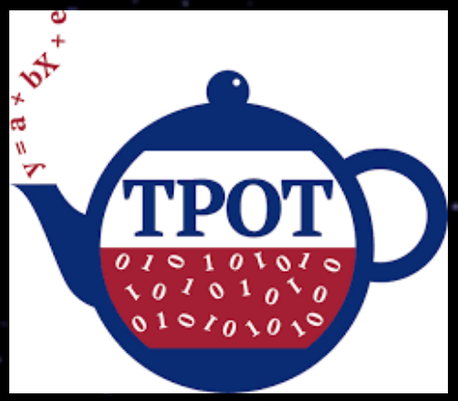
EXPLOSION OF COMMERCIAL INTEREST



(MicroSoft Azure)

EXPLOSION OF FUNDAMENTAL RESEARCH, HIGH-QUALITY OPEN SOURCE TOOLS

mlrMBO



Featuretools

Auto-Sklearn

hyperband

OpenML

We'll focus on these (and the concepts behind them)

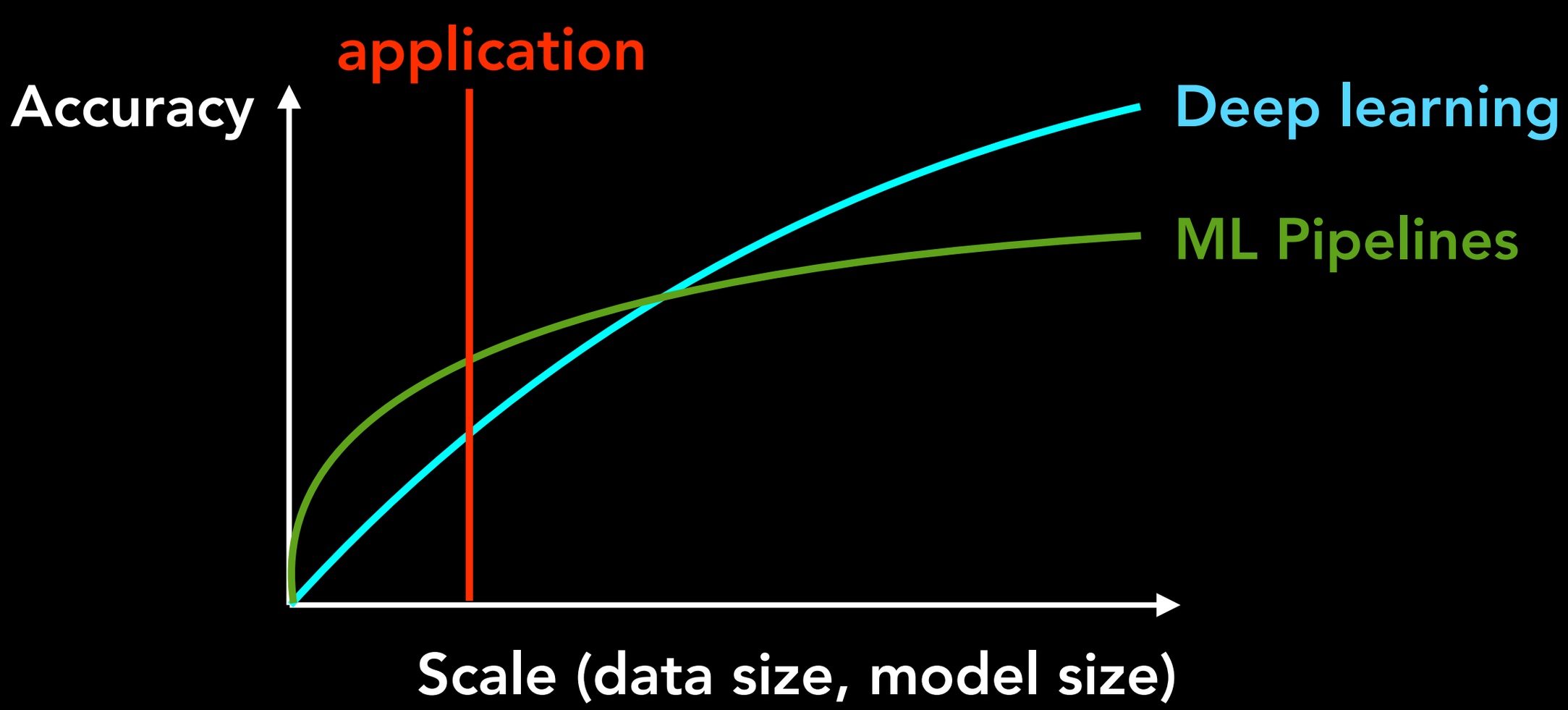
AUTOMATIC MACHINE LEARNING

Not about automating data scientists

- **Efficient** exploration of techniques
  - Automate the tedious aspects (inner loop)
  - Make every data scientist a super data scientist
- **Democratisation**
  - Allow individuals, small companies to use machine learning effectively (at lower cost)
  - Open source tools and platforms
- Data **Science**
  - Better understand algorithms, develop better ones
  - Self-learning algorithms

AUTOMATIC MACHINE LEARNING

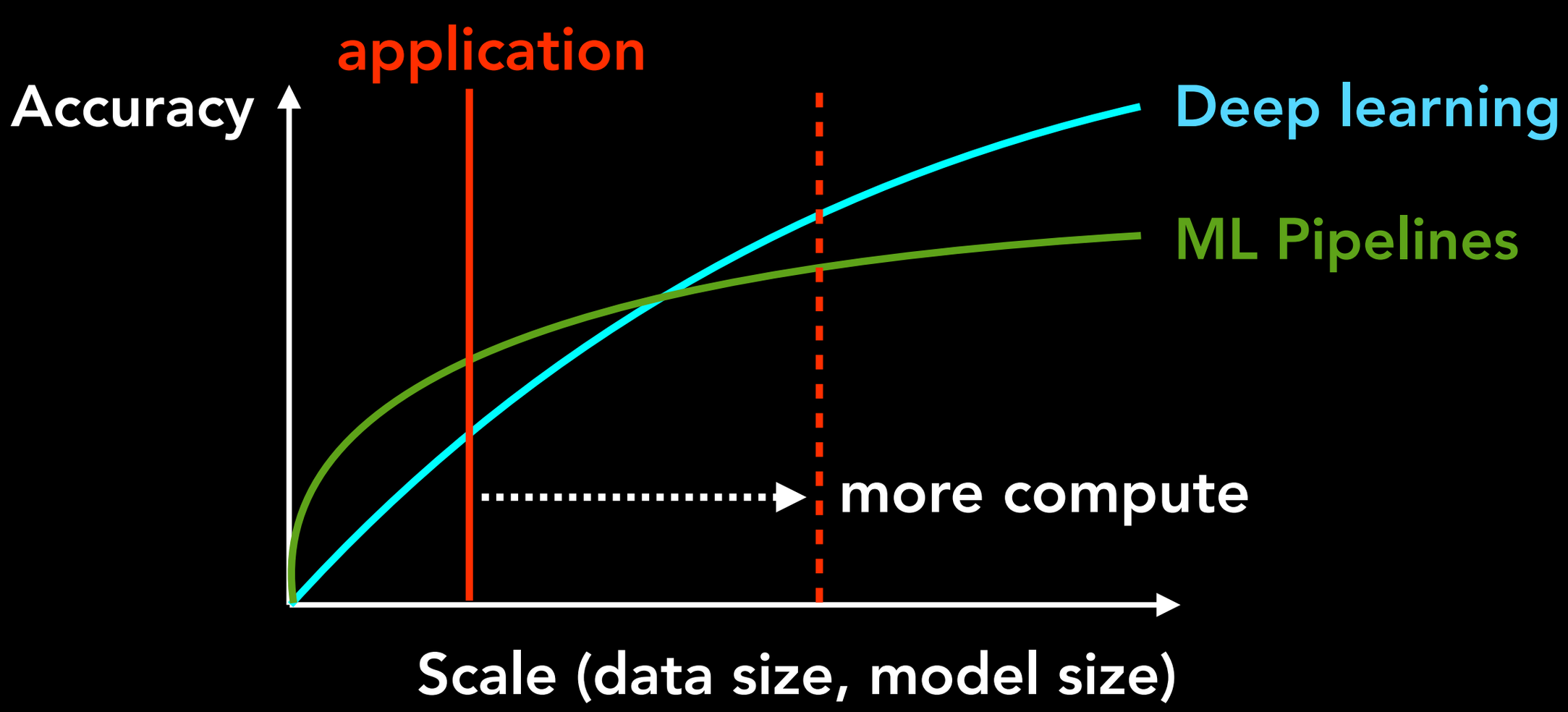
- ***End-to-end***: depending on the application, this can be a (complex) **machine learning pipeline**, a **deep learning architecture**, or a combination



Graph by Olivier Bousquet

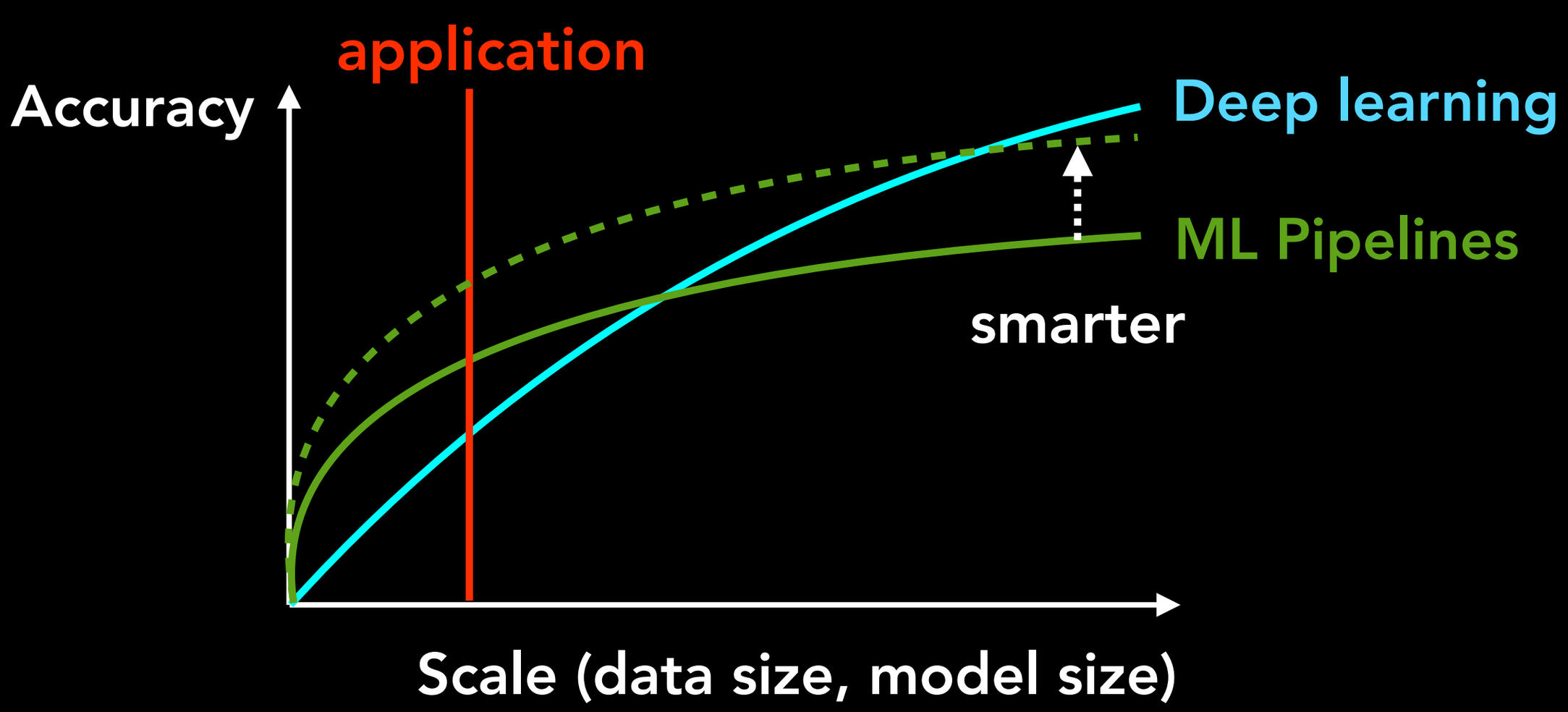
AUTOMATIC MACHINE LEARNING

- ***End-to-end***: depending on the application, this can be a (complex) **machine learning pipeline**, a **deep learning architecture**, or a combination



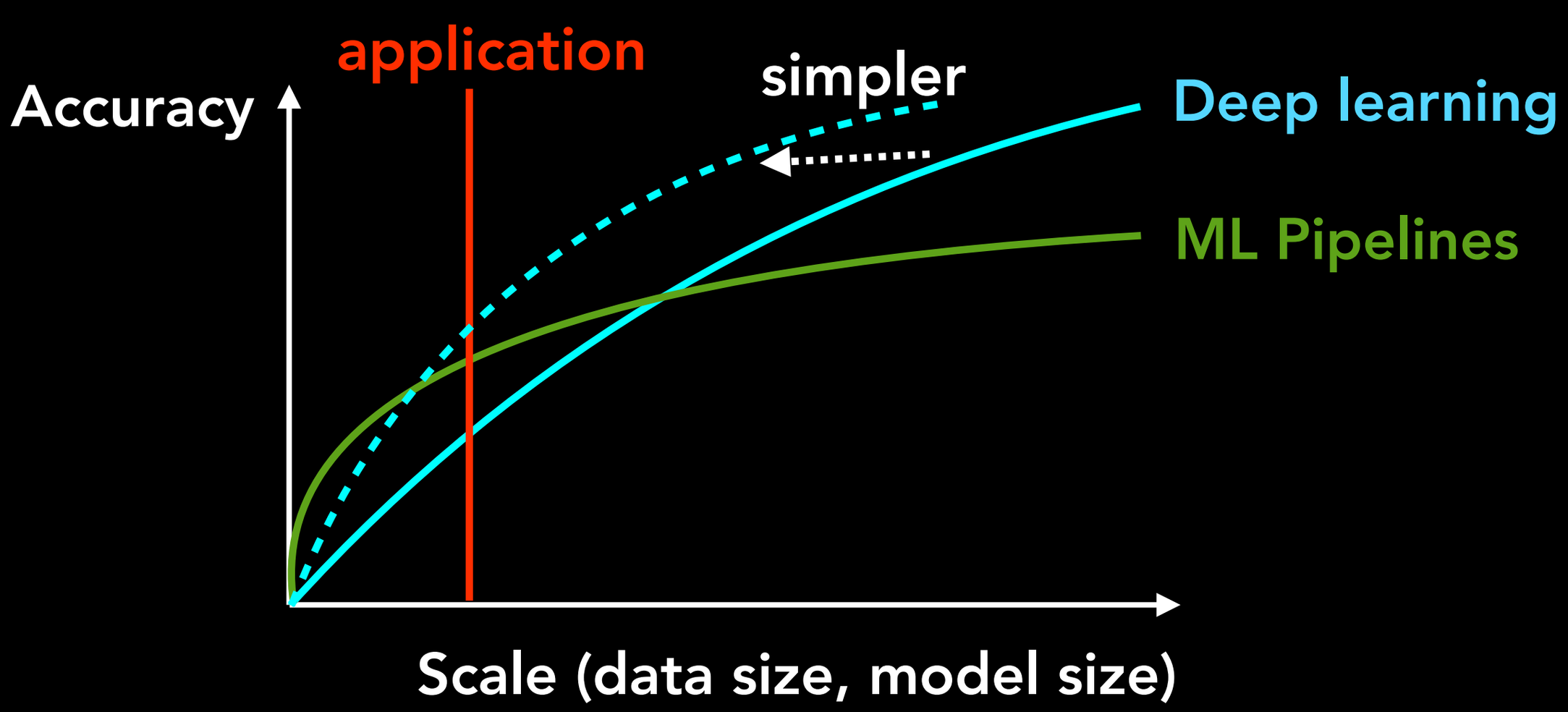
AUTOMATIC MACHINE LEARNING

- (Semi)automatically design **better pipelines**
  - Combine preprocessing techniques, feature selection / creation, model selection, tuning,…

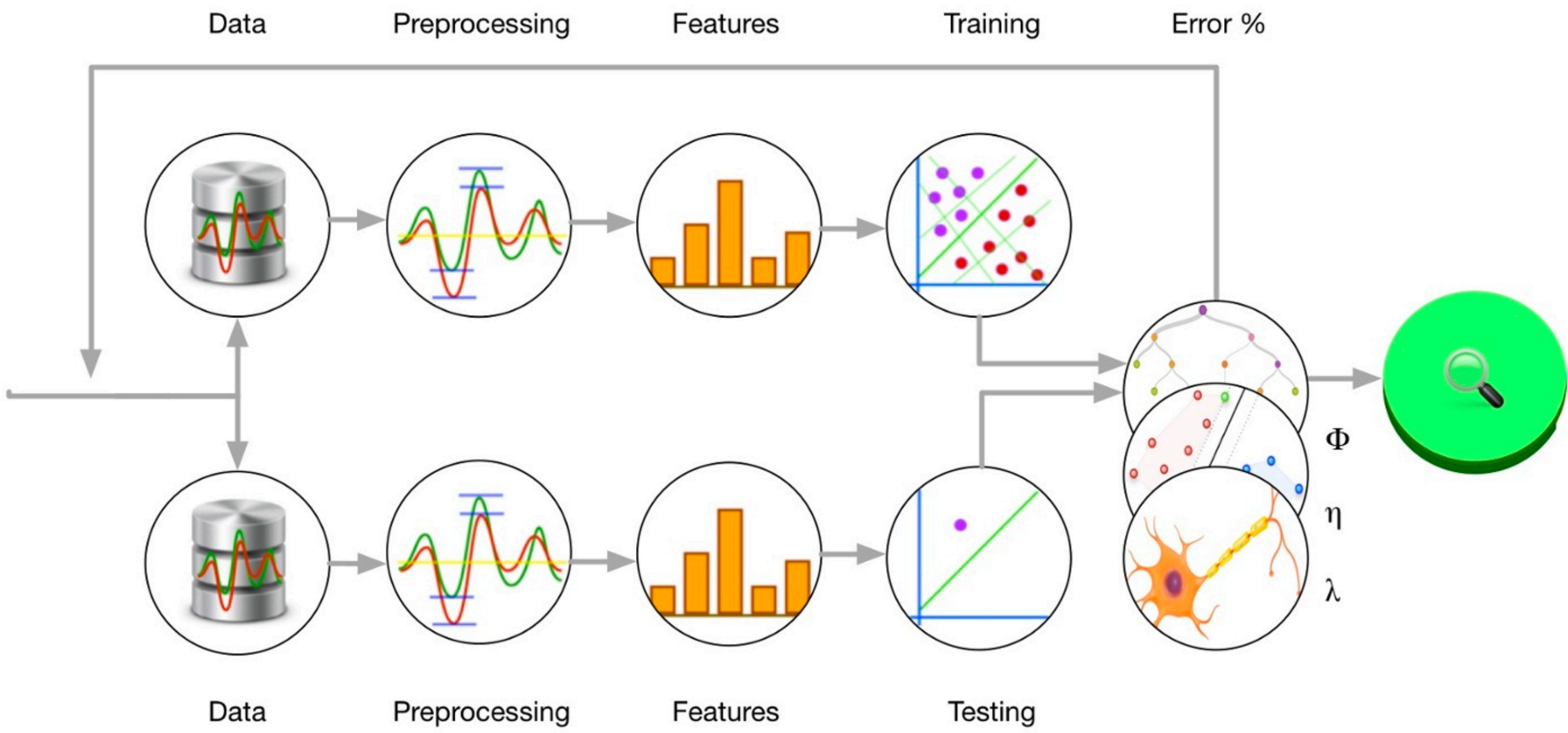


AUTOMATIC MACHINE LEARNING

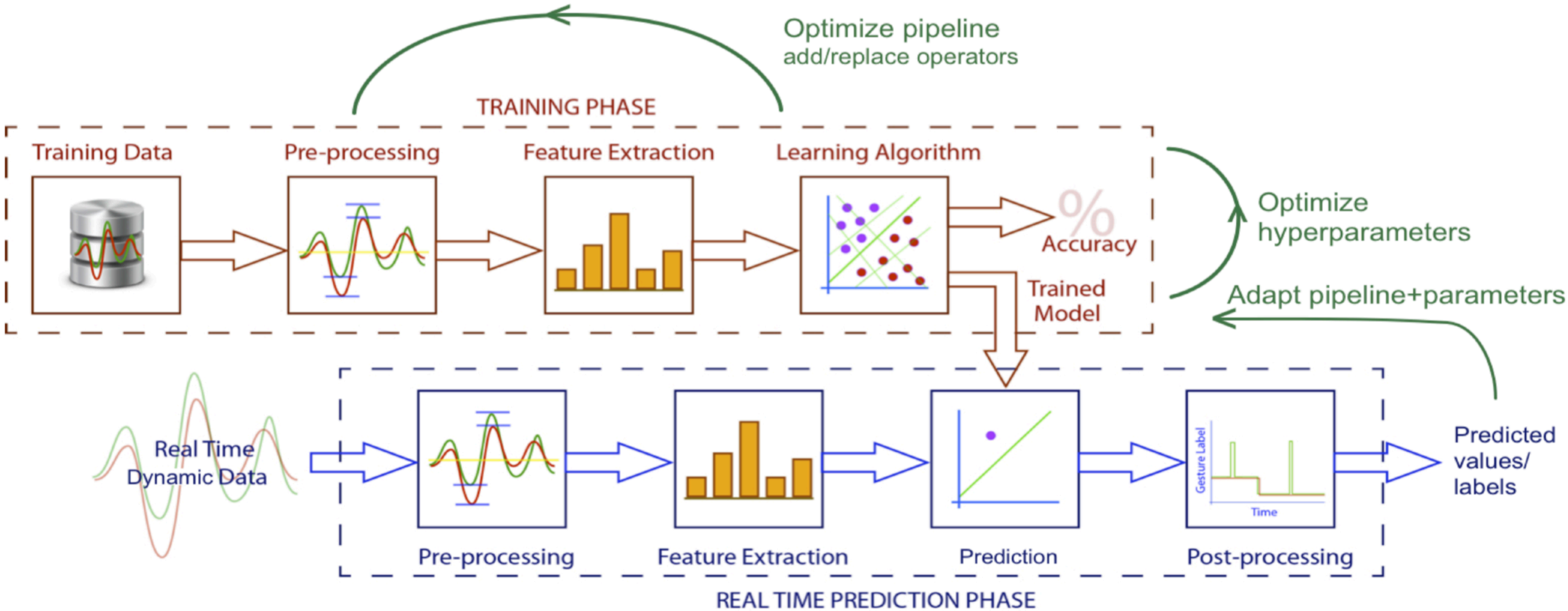
- (Semi)automatically design **better architectures**
  - Neural architecture search, tuning of optimization algorithms,…



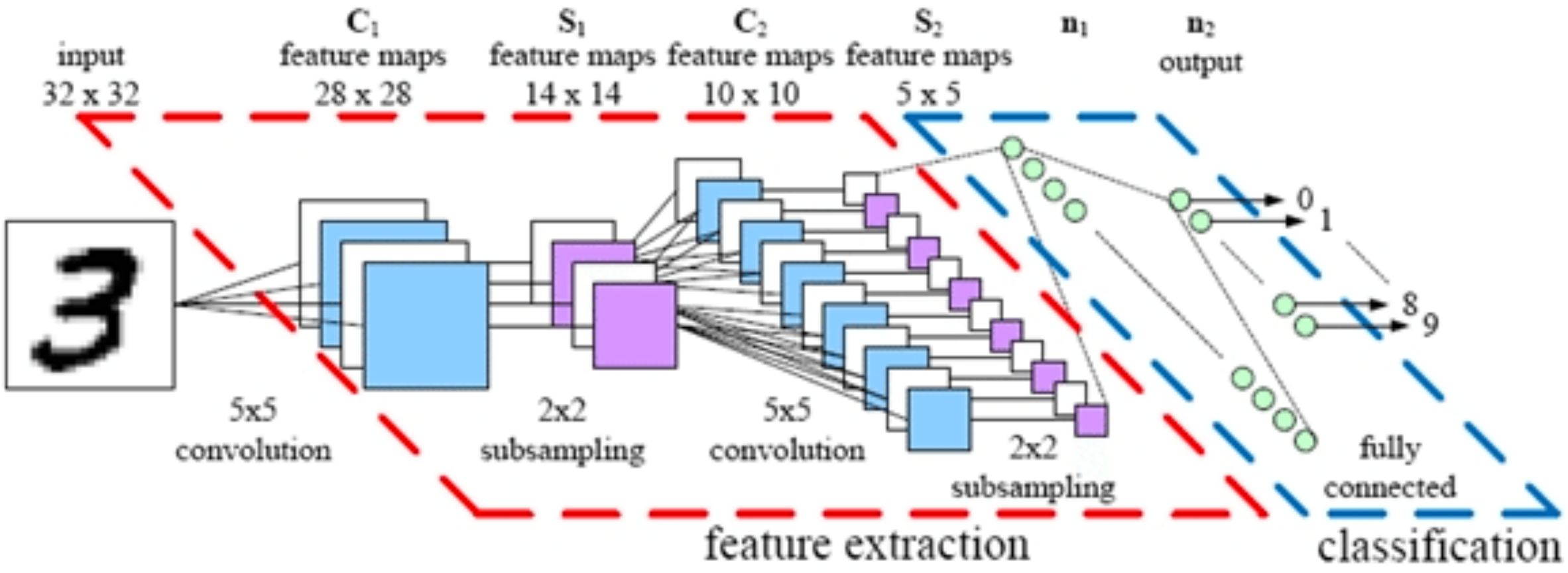
MACHINE LEARNING PIPELINES



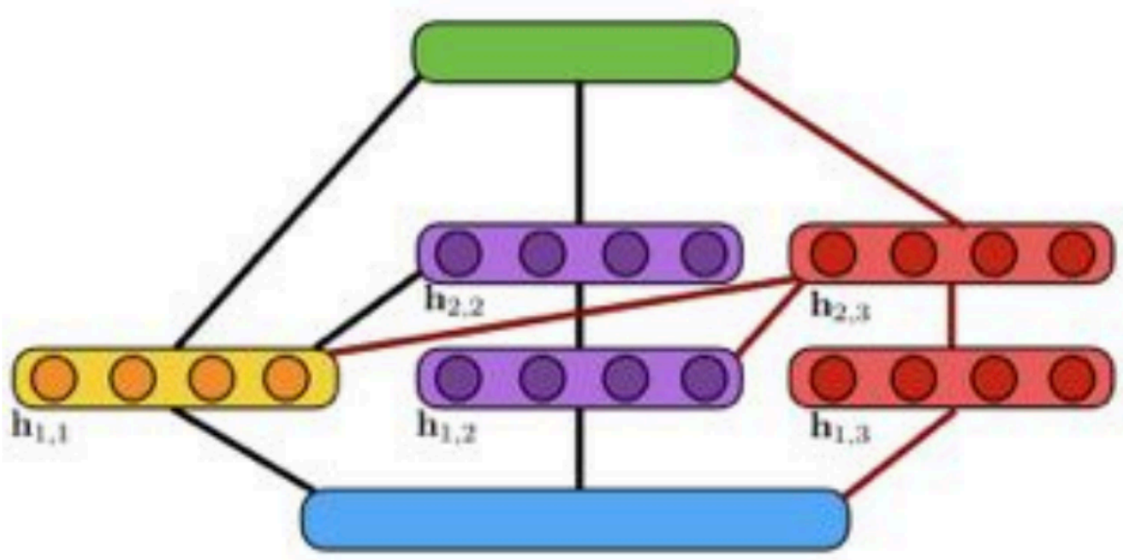
AUTOMATING MACHINE LEARNING PIPELINES

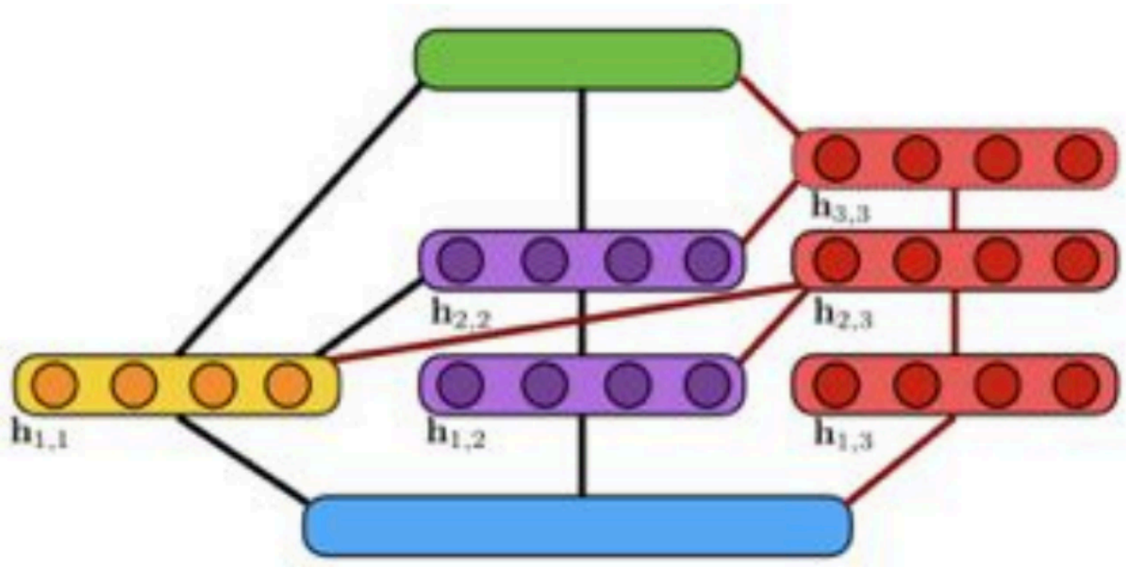


NEURAL ARCHITECTURES



NEURAL ARCHITECTURE SEARCH





AdaNet (Google)

HYPERPARAMETERS

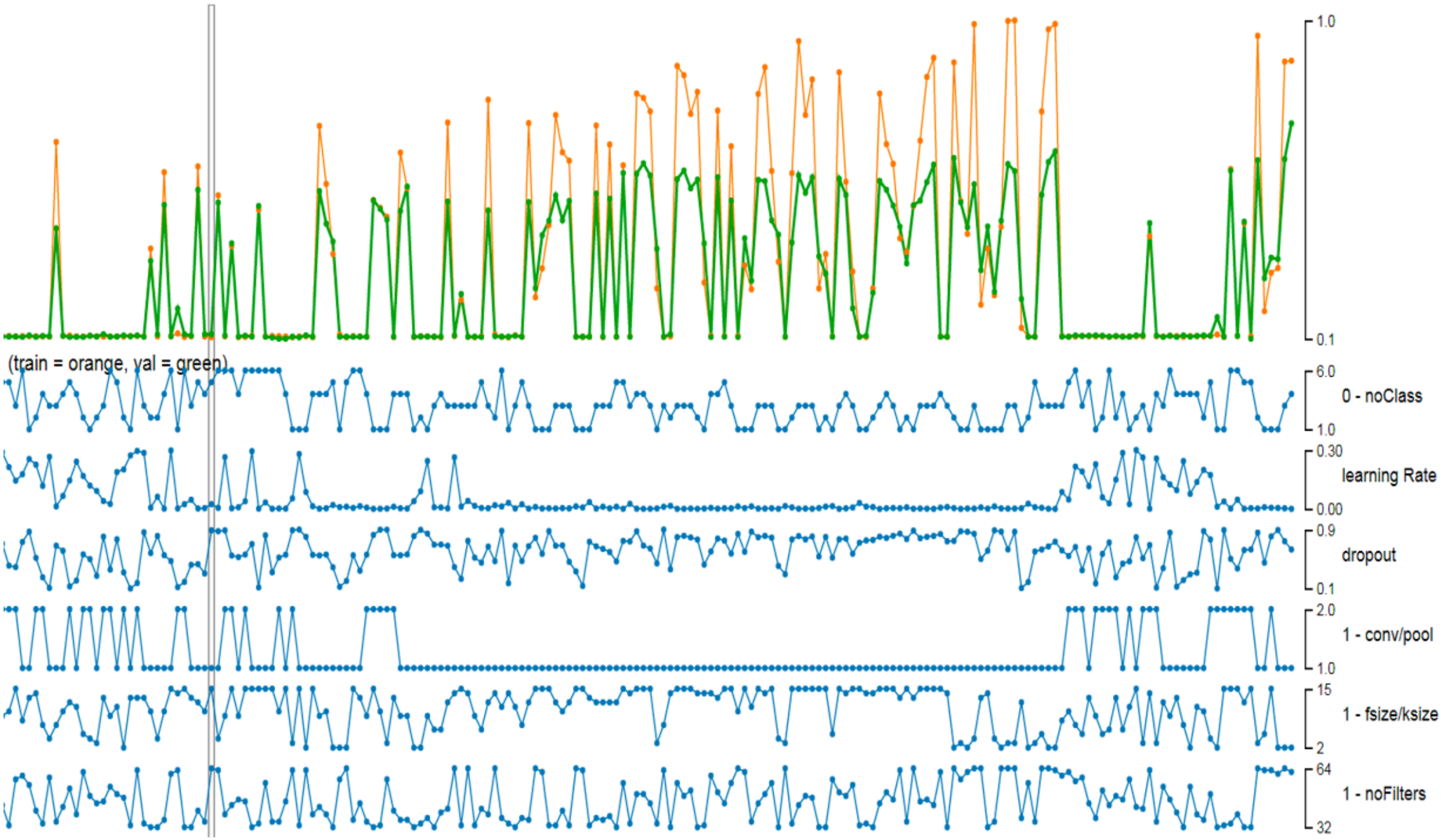
Everything user has to decide before running the algorithm

Different types, ranges, dependencies. E.g. neural net:

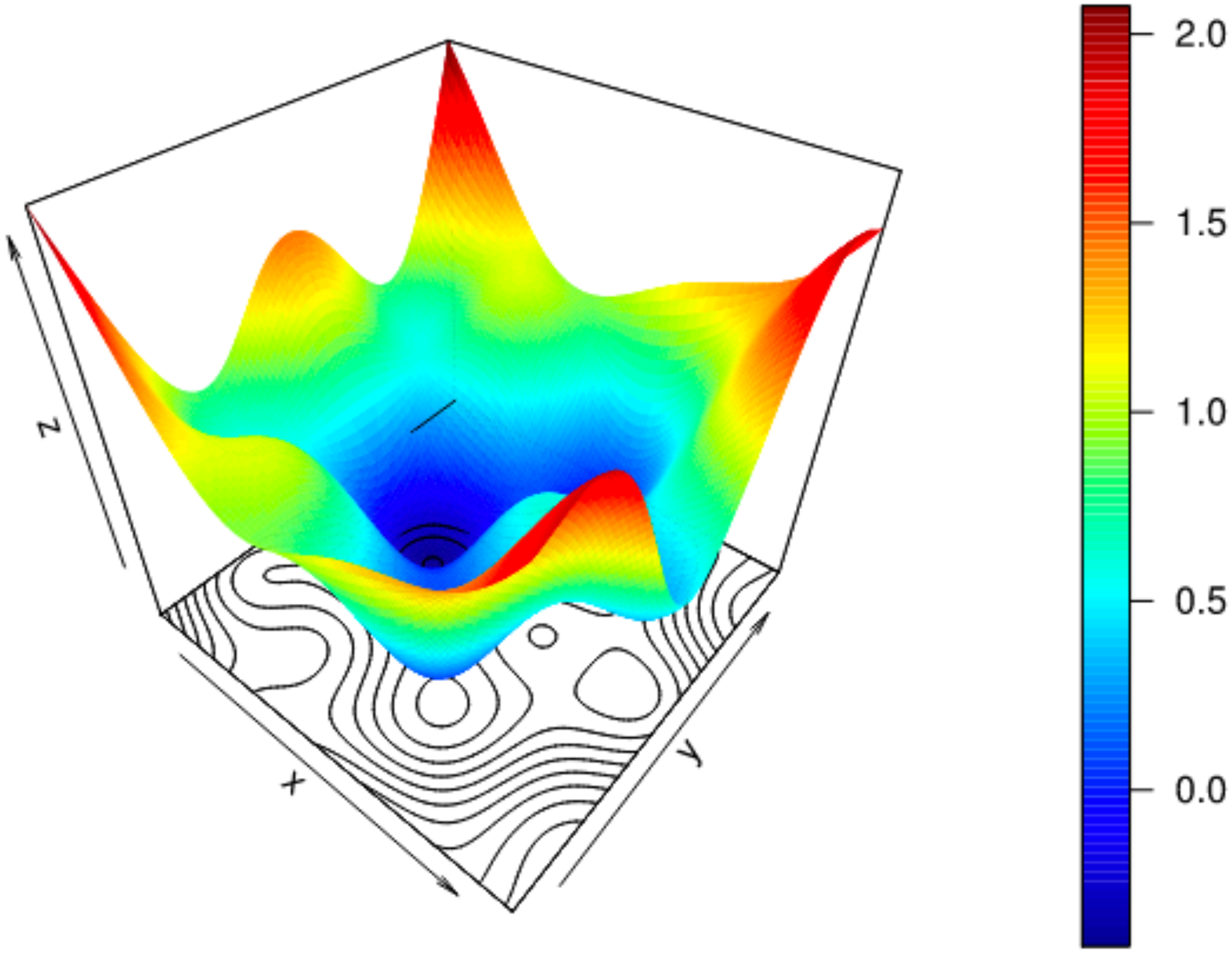
	Name	Range	Default	log scale	Type	Conditional
Network hyperparameters	batch size	$[32, 4096]$	32	✓	float	-
	number of updates	$[50, 2500]$	200	✓	int	-
	number of layers	$[1, 6]$	1	-	int	-
	learning rate	$[10^{-6}, 1.0]$	10^{-2}	✓	float	-
	L_2 regularization	$[10^{-7}, 10^{-2}]$	10^{-4}	✓	float	-
	dropout output layer	$[0.0, 0.99]$	0.5	✓	float	-
	solver type	{SGD, Momentum, Adam, Adadelta, Adagrad, smorm, Nesterov }	smorm3s	-	cat	-
	lr-policy	{Fixed, Inv, Exp, Step}	fixed	-	cat	-
Conditioned on solver type	β_1	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	β_2	$[10^{-4}, 10^{-1}]$	10^{-1}	✓	float	✓
	ρ	$[0.05, 0.99]$	0.95	✓	float	✓
	momentum	$[0.3, 0.999]$	0.9	✓	float	✓
Conditioned on lr-policy	γ	$[10^{-3}, 10^{-1}]$	10^{-2}	✓	float	✓
	k	$[0.0, 1.0]$	0.5	-	float	✓
	s	$[2, 20]$	2	-	int	✓
Per-layer hyperparameters	activation-type	{Sigmoid, TanH, ScaledTanH, ELU, ReLU, Leaky, Linear}	ReLU	-	cat	✓
	number of units	$[64, 4096]$	128	✓	int	✓
	dropout in layer	$[0.0, 0.99]$	0.5	-	float	✓
	weight initialization	{Constant, Normal, Uniform, Glorot-Uniform, Glorot-Normal, He-Normal, He-Uniform, Orthogonal, Sparse}	He-Normal	-	cat	✓
	std. normal init.	$[10^{-7}, 0.1]$	0.0005	-	float	✓
	leakiness	$[0.01, 0.99]$	$\frac{1}{3}$	-	float	✓
	tanh scale in	$[0.5, 1.0]$	$2/3$	-	float	✓
	tanh scale out	$[1.1, 3.0]$	1.7159	✓	float	✓

HYPERPARAMETERS

Complex interactions, different effects on performance



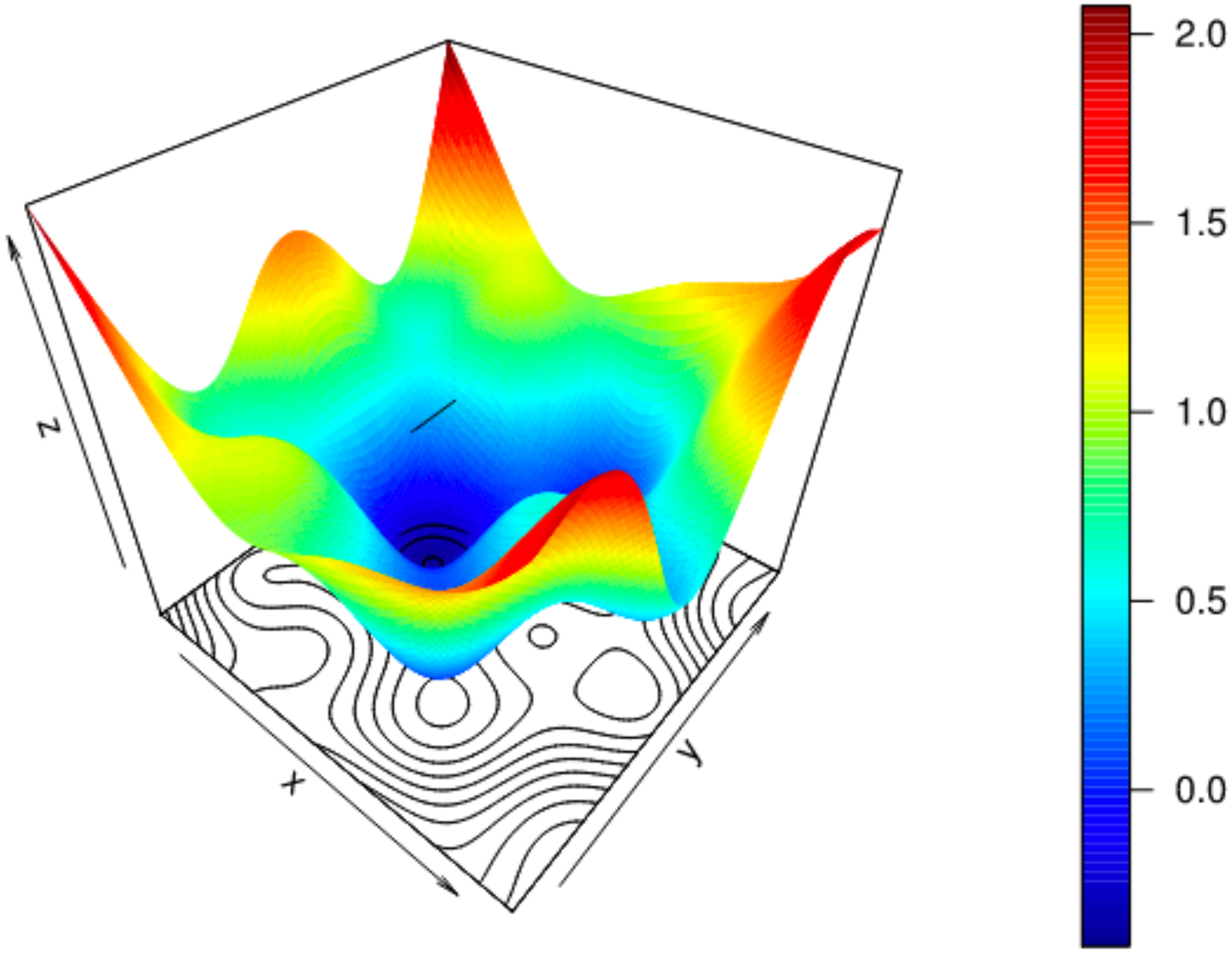
HYPERPARAMETER SPACE



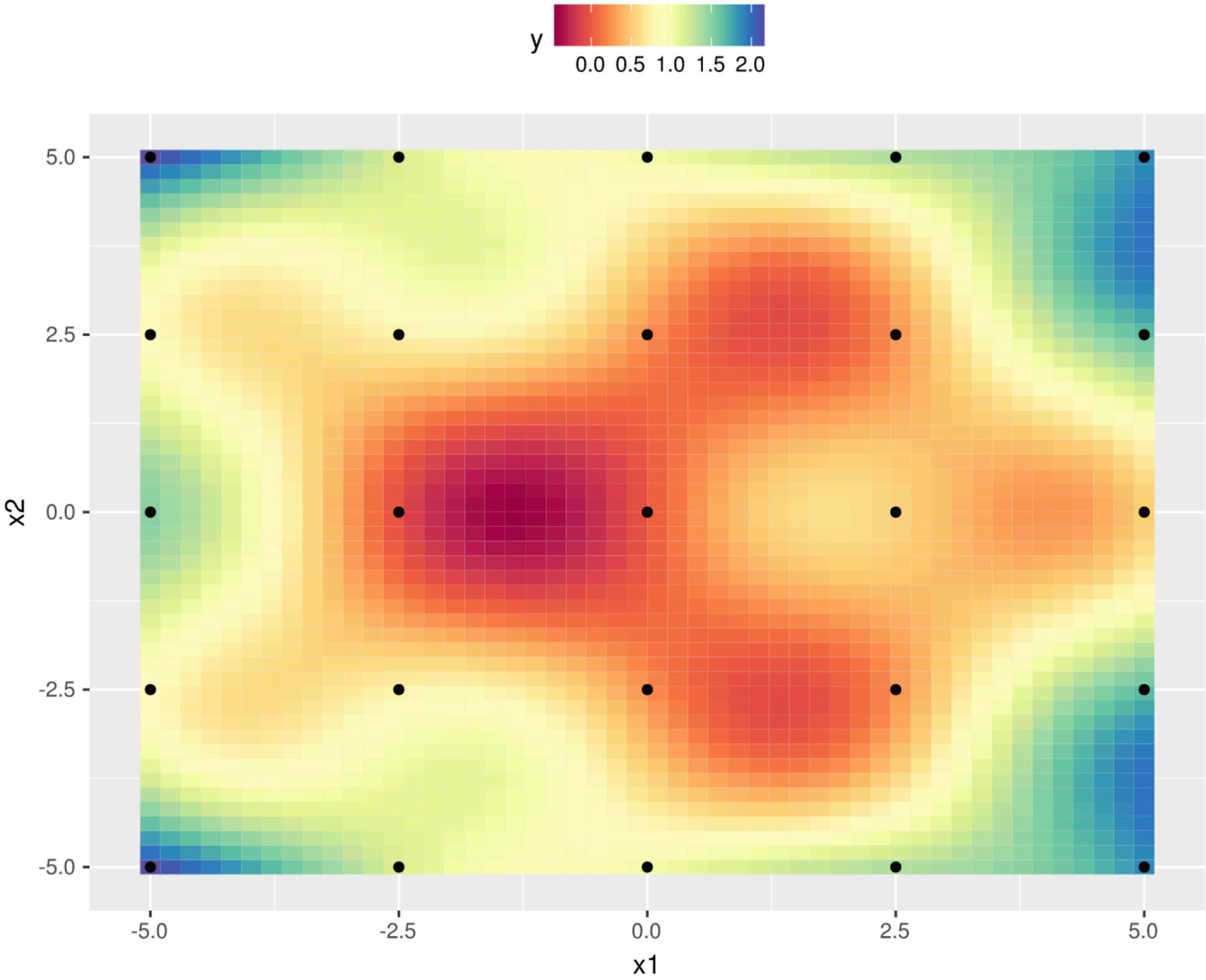
BLACK BOX OPTIMIZATION

- Typically, represent the AutoML problem as a large n-dimensional (Euclidean) search space: configuration is a point in that space

HYPERPARAMETER SPACE (2D)

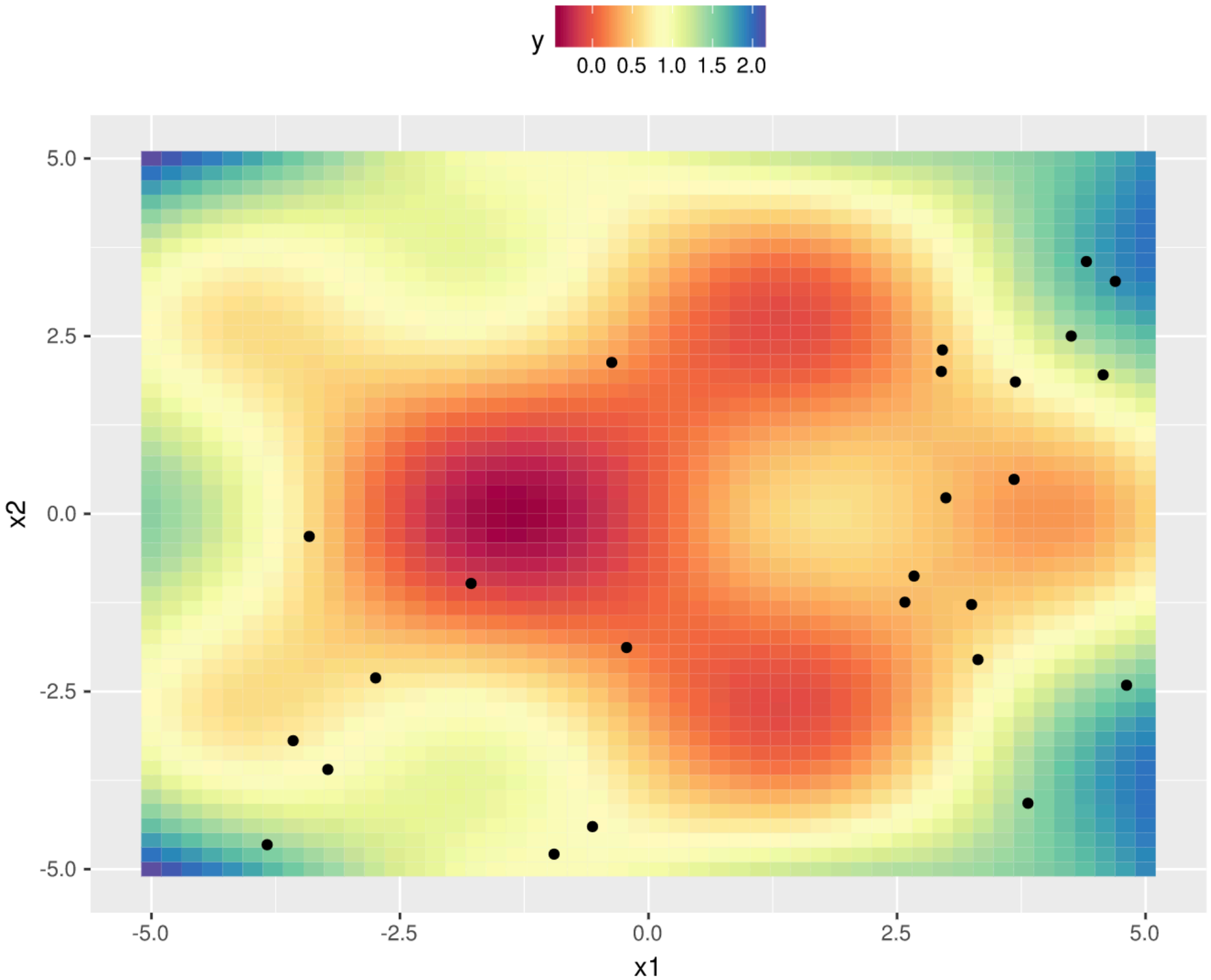


HYPERPARAMETER SPACE



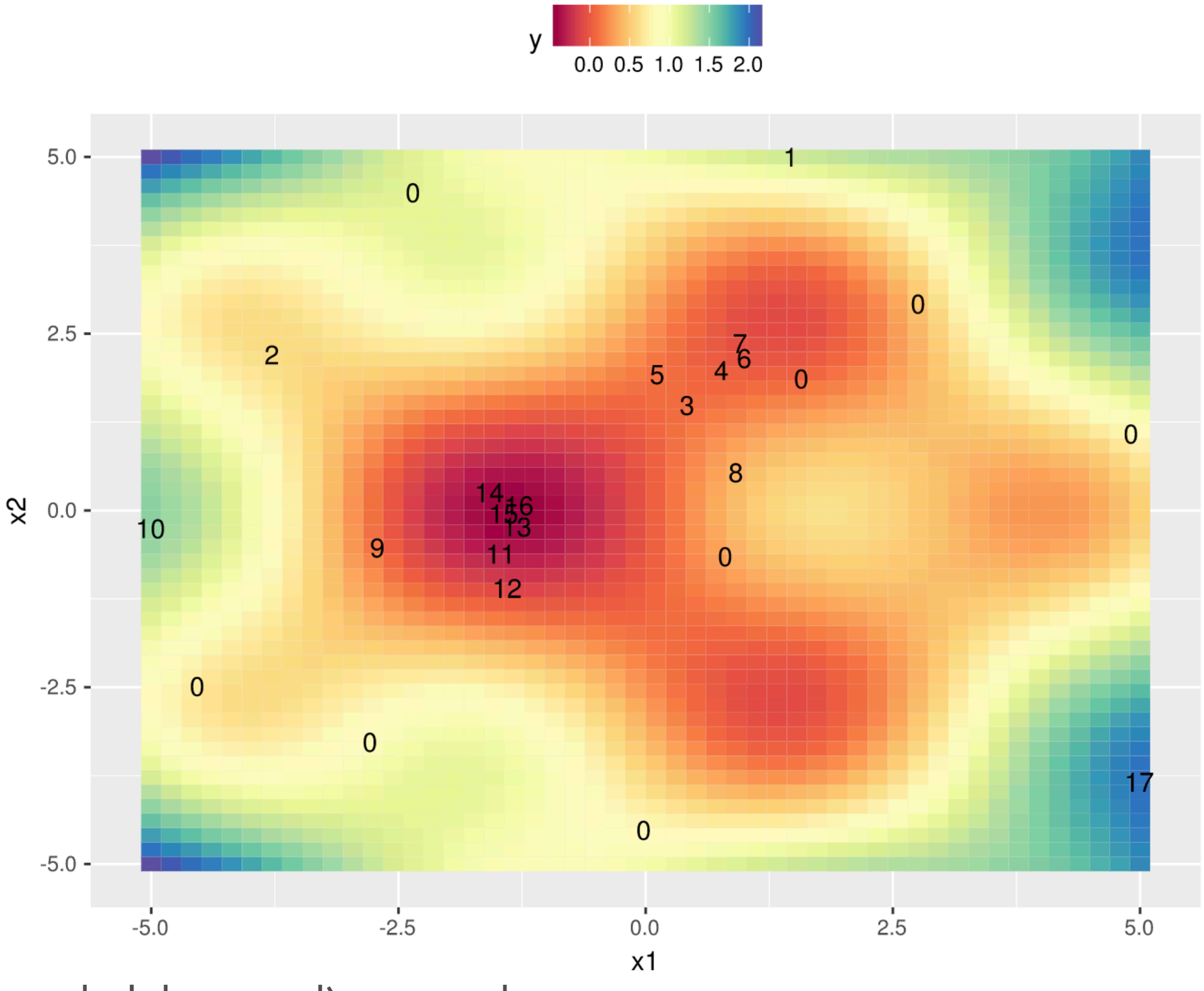
Grid search

HYPERPARAMETER SPACE



Random search

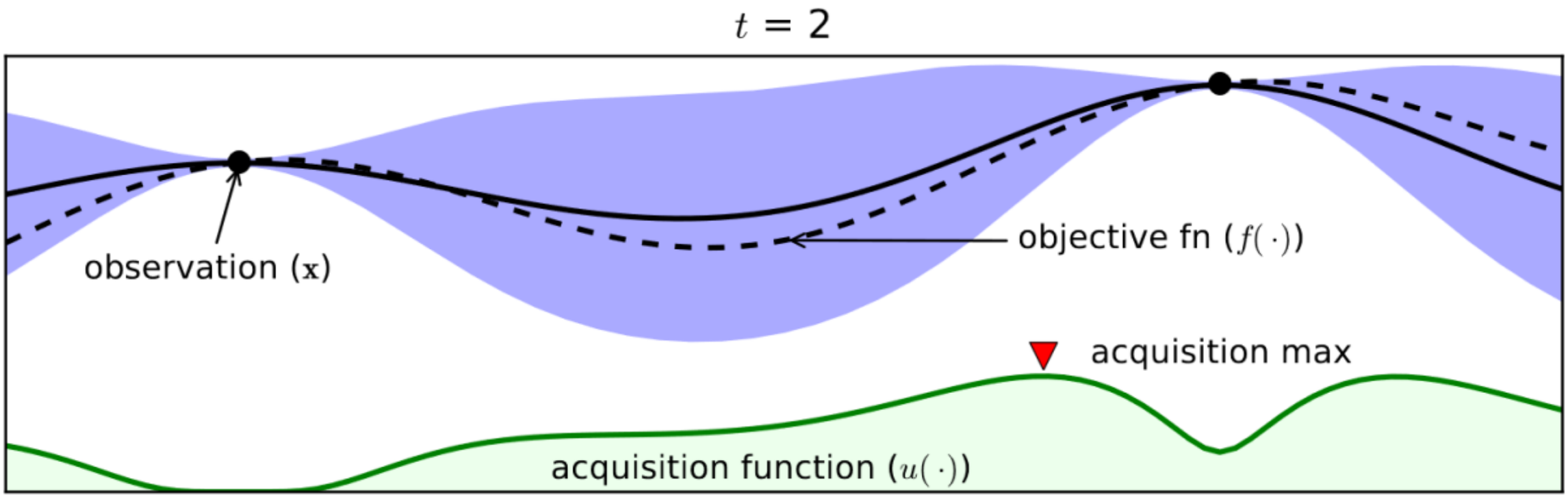
HYPERPARAMETER SPACE



Guided (model-based) search

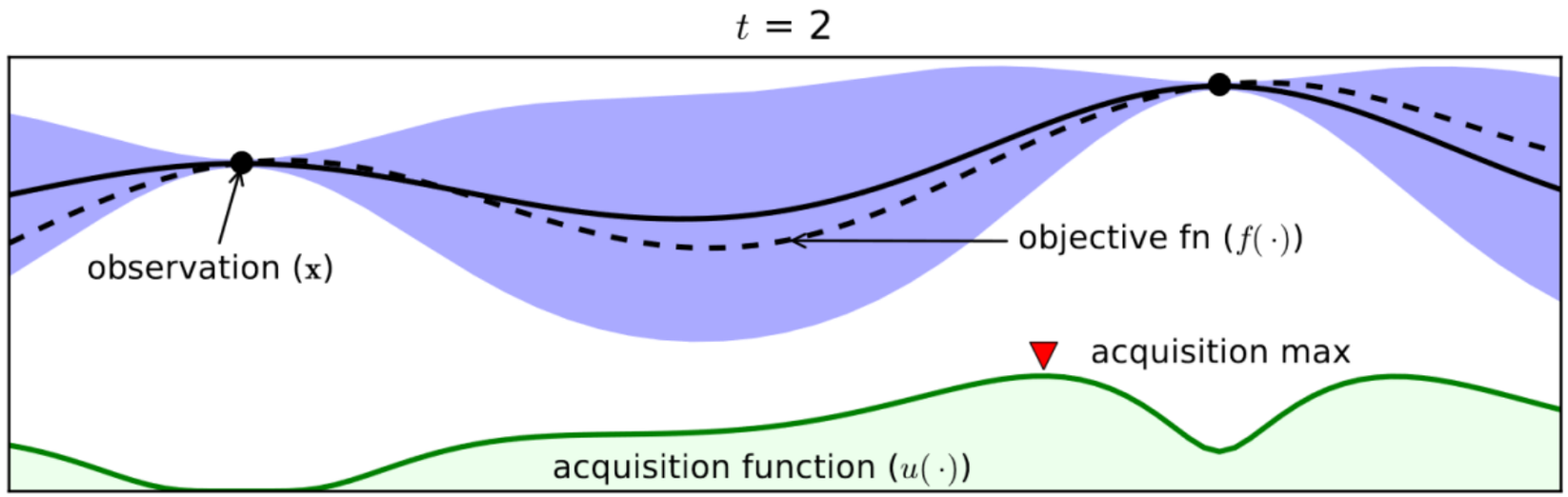
BAYESIAN OPTIMIZATION

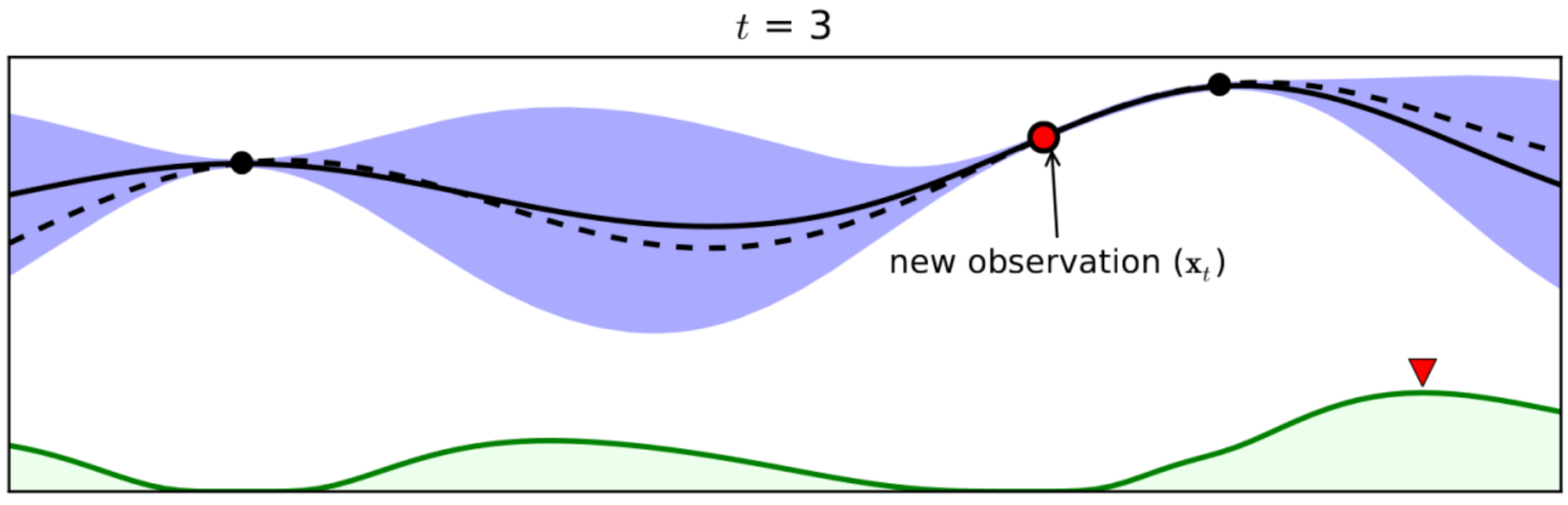
- Start with a few (random) hyperparameter configurations
- Build a ***surrogate model*** to predict how well other configurations will work (probabilistic, regression)
  - Most popular: Gaussian processes, Random Forests
- To avoid a greedy search, use an ***acquisition function*** to trade off exploration and exploitation.
- The next configuration is the best one under that function

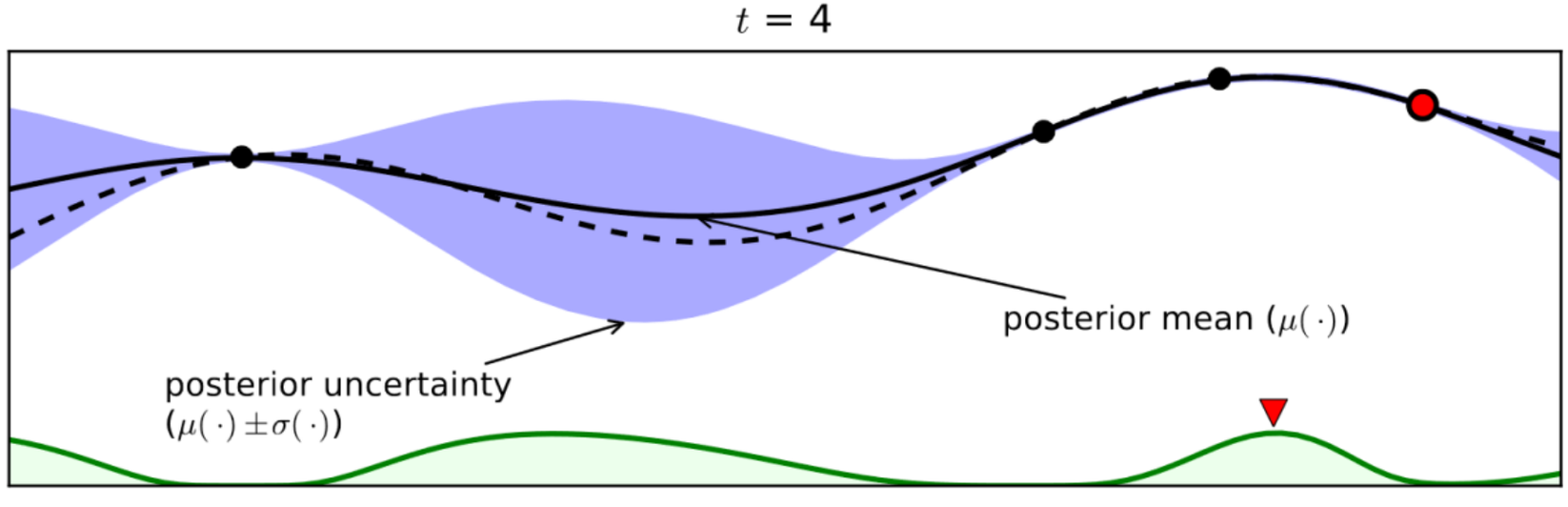


BAYESIAN OPTIMIZATION

- Repeat
- Stopping criterion:
  - Fixed budget (time, evaluations)
  - Min. distance between configs
  - Threshold for acquisition function
  - …
- Overfits quite easily



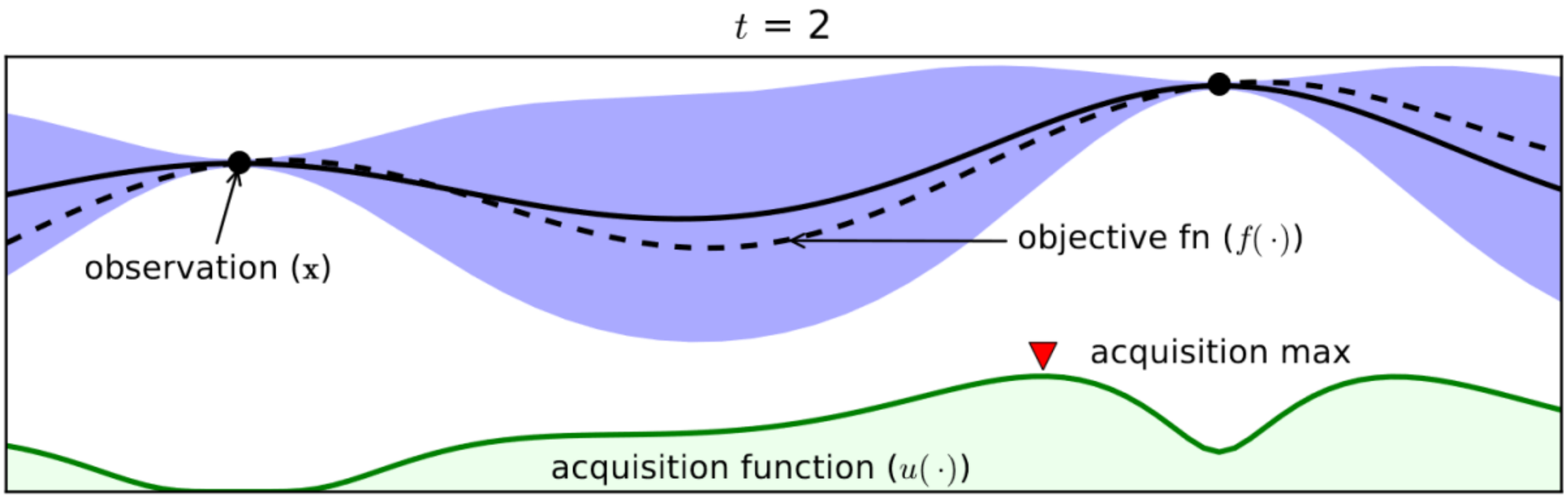




https://raw.githubusercontent.com/mlr-org/mlrMBO/master/docs/articles/helpers/animation-.gif

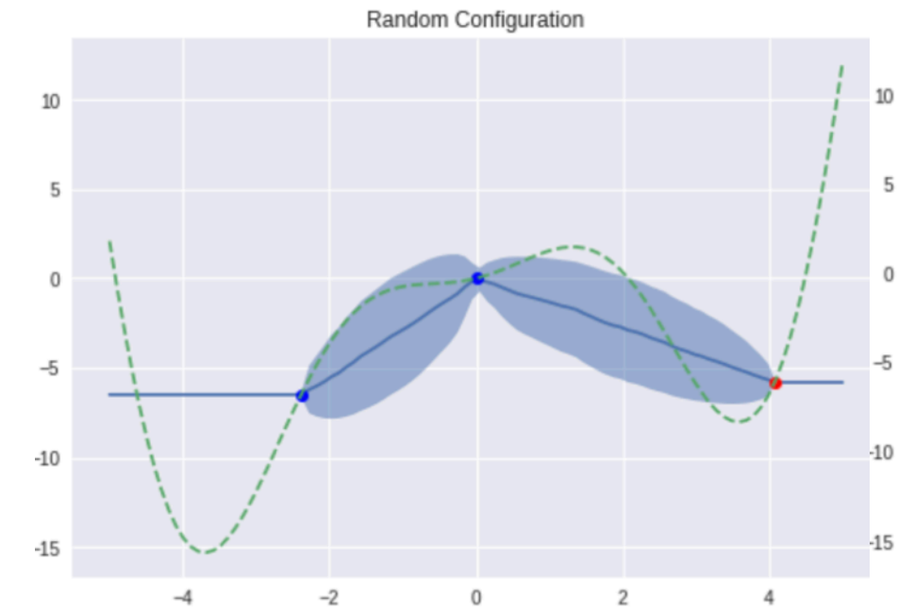
SURROGATE MODELS

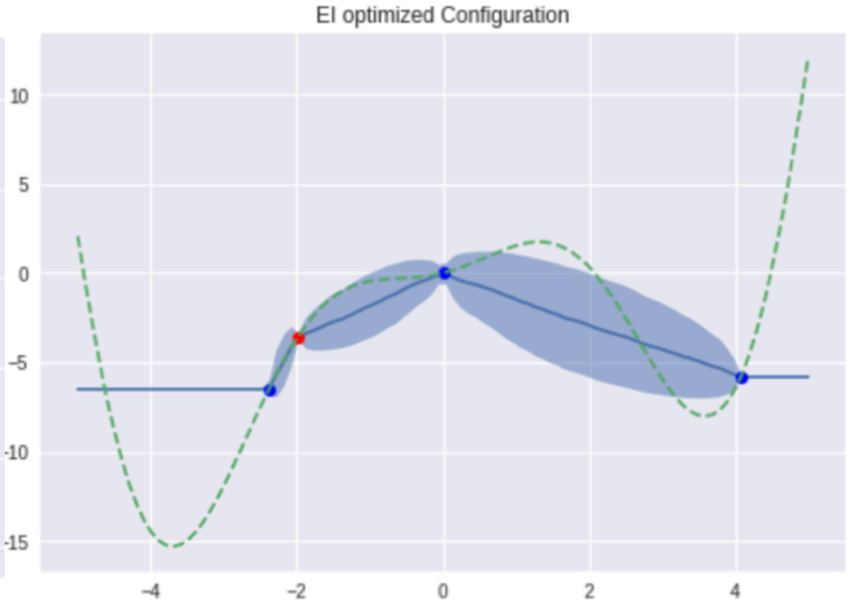
- Lot of research into surrogate models:
  - Gaussian processes: extrapolate well, not scalable
    - Sparse GPs (select ‘inducing points’ that minimize info loss)
    - Multi-task GPs (transfer surrogate models from other tasks)
  - RandomForest: more scalable, extrapolates badly
  - Bayesian Neural Networks (expensive, sensitive to hyperparams)
  - ...

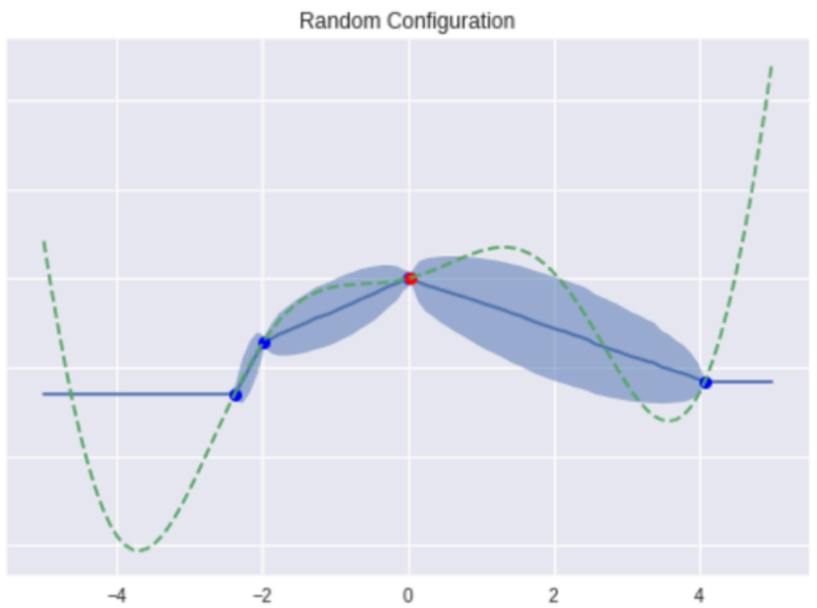


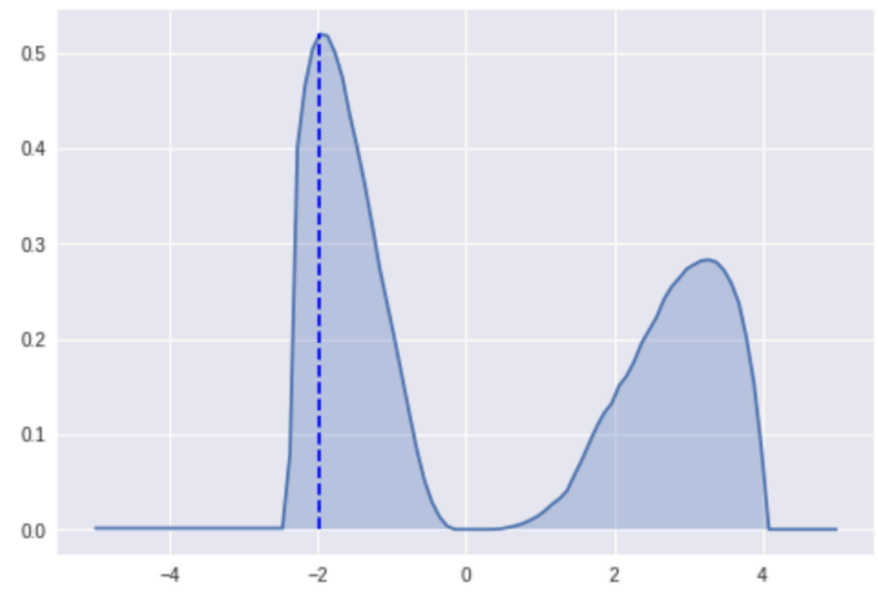
SURROGATE MODELS

- SMAC: Random Forests (with random configs in between):

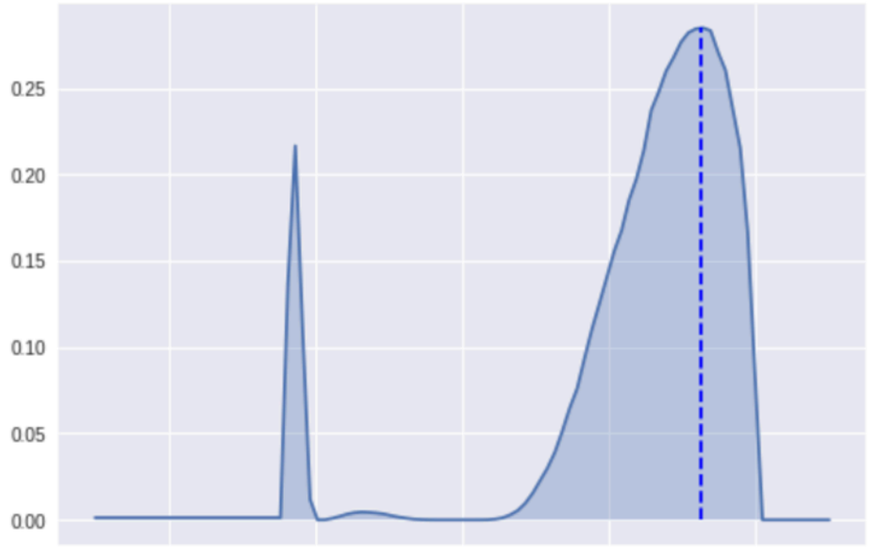






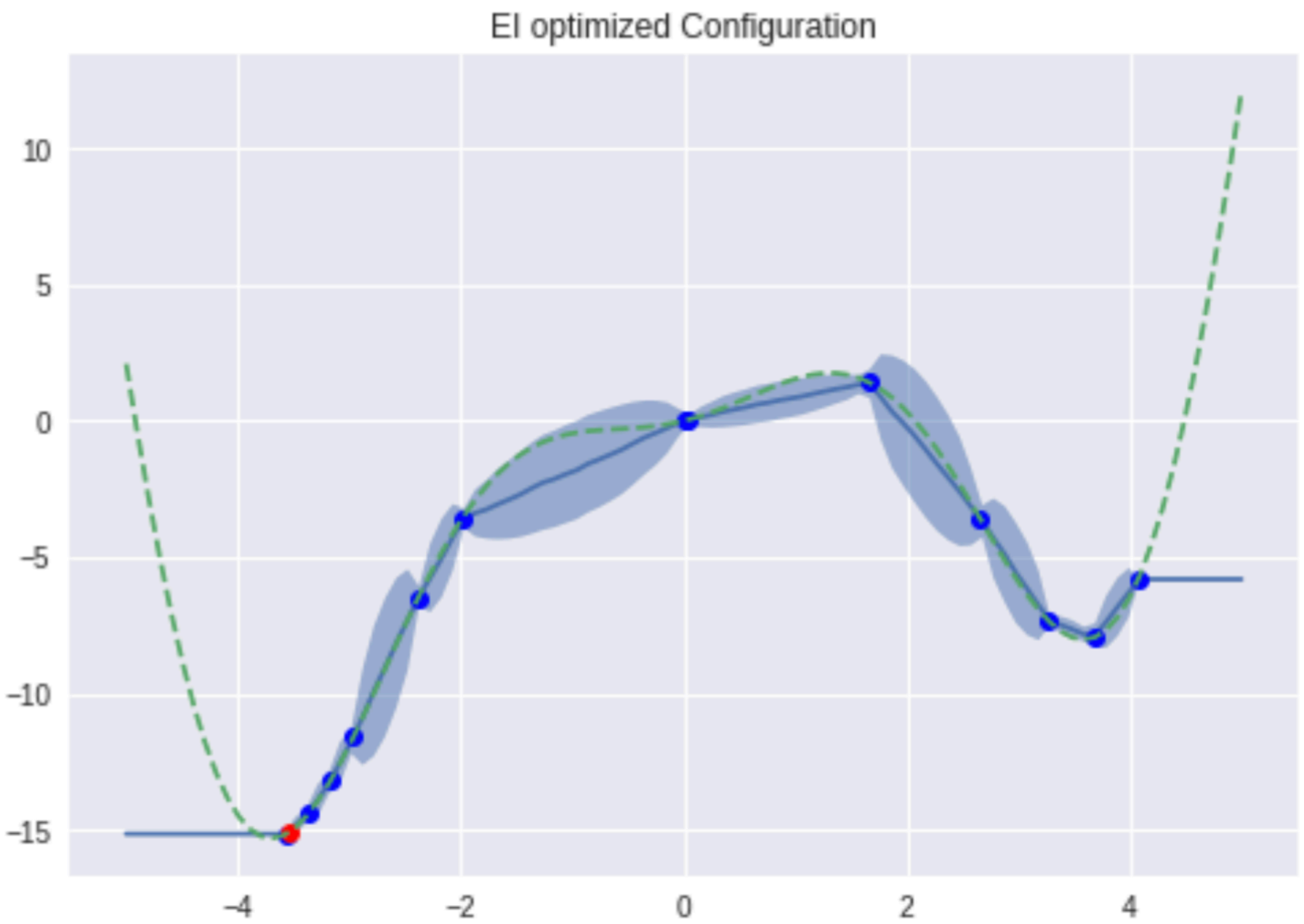


(minimizing)

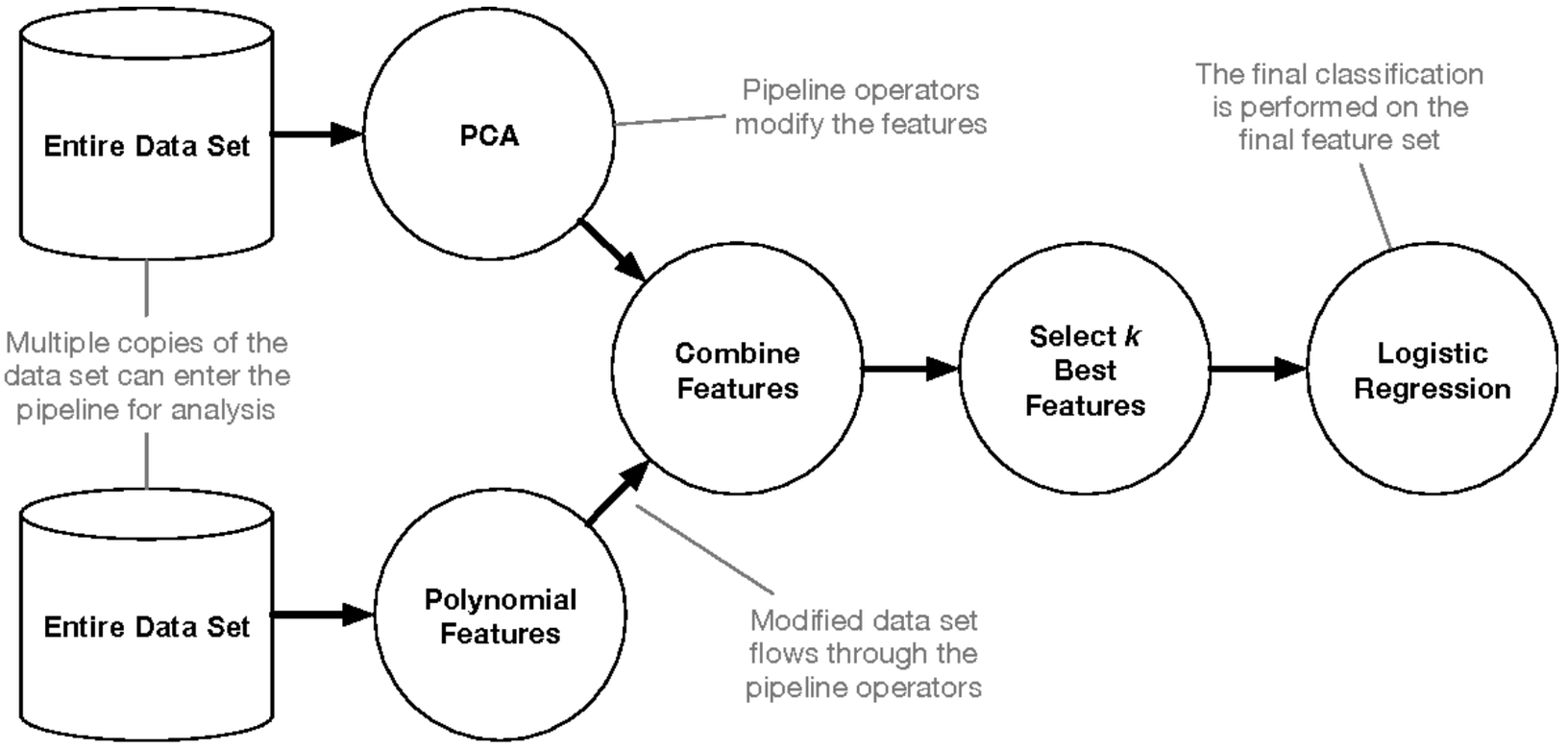


SURROGATE MODELS

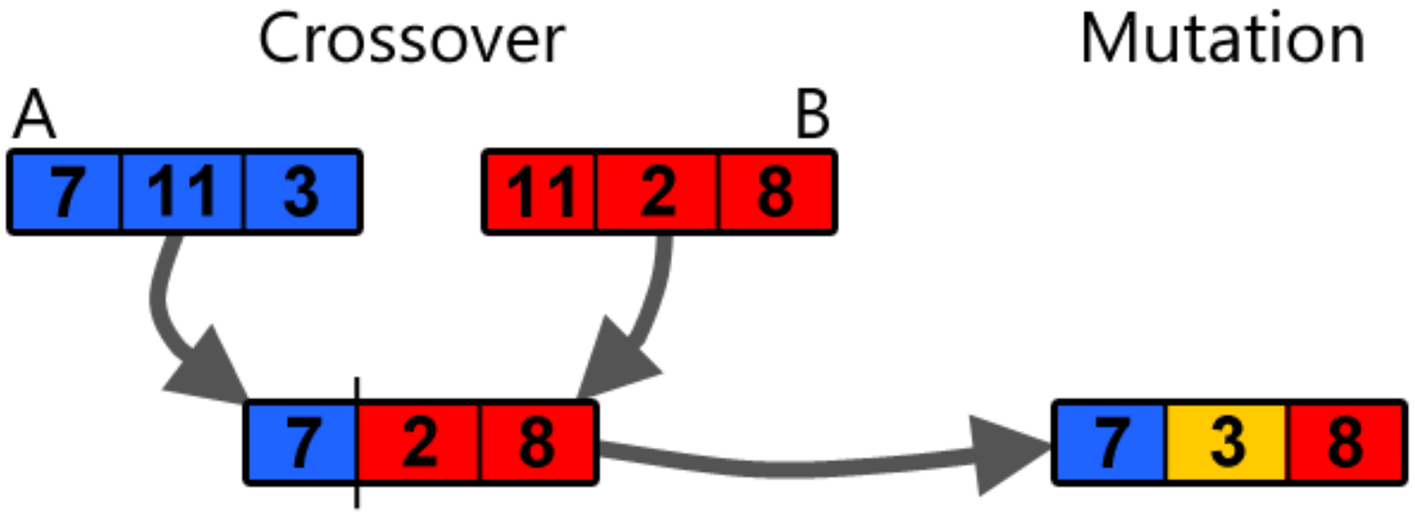
- SMAC: Random Forests (with random configs in between):
  - After 12 iterations: extrapolation remains an issue



GENETIC PROGRAMMING



Randall Olson et al. (2016) GECCO



GENETIC PROGRAMMING

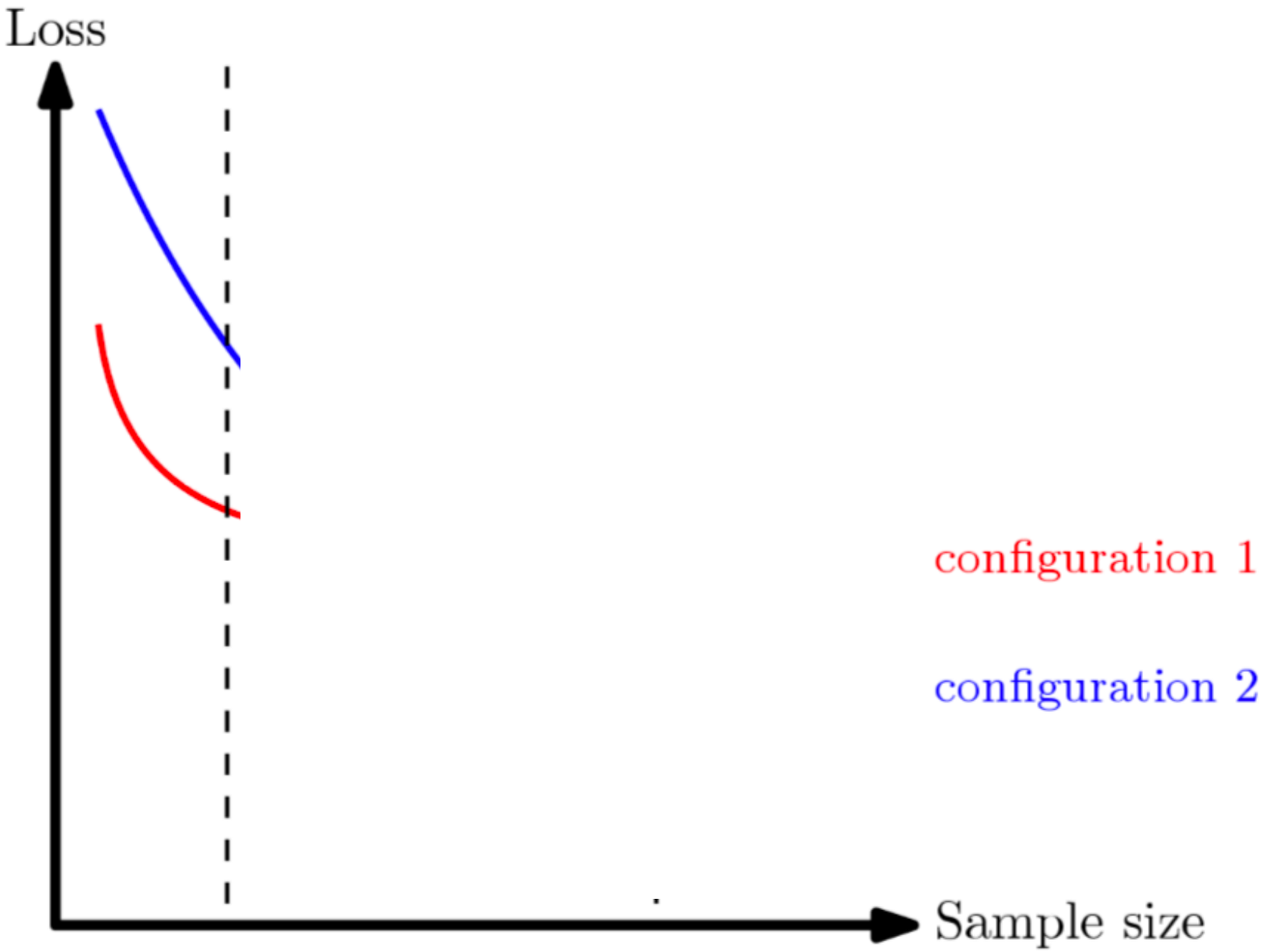
```
from tpot import TPOTClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target,
                                                    train_size=0.75, test_size=0.25)

tpot = TPOTClassifier(generations=5, population_size=50, verbosity=2, n_jobs=-1)
tpot.fit(X_train, y_train)
```

```
Optimization Progress:   0%|          | 0/300 [00:00<?, ?pipeline/s]
```

```
print(tpot.score(X_test, y_test))
```

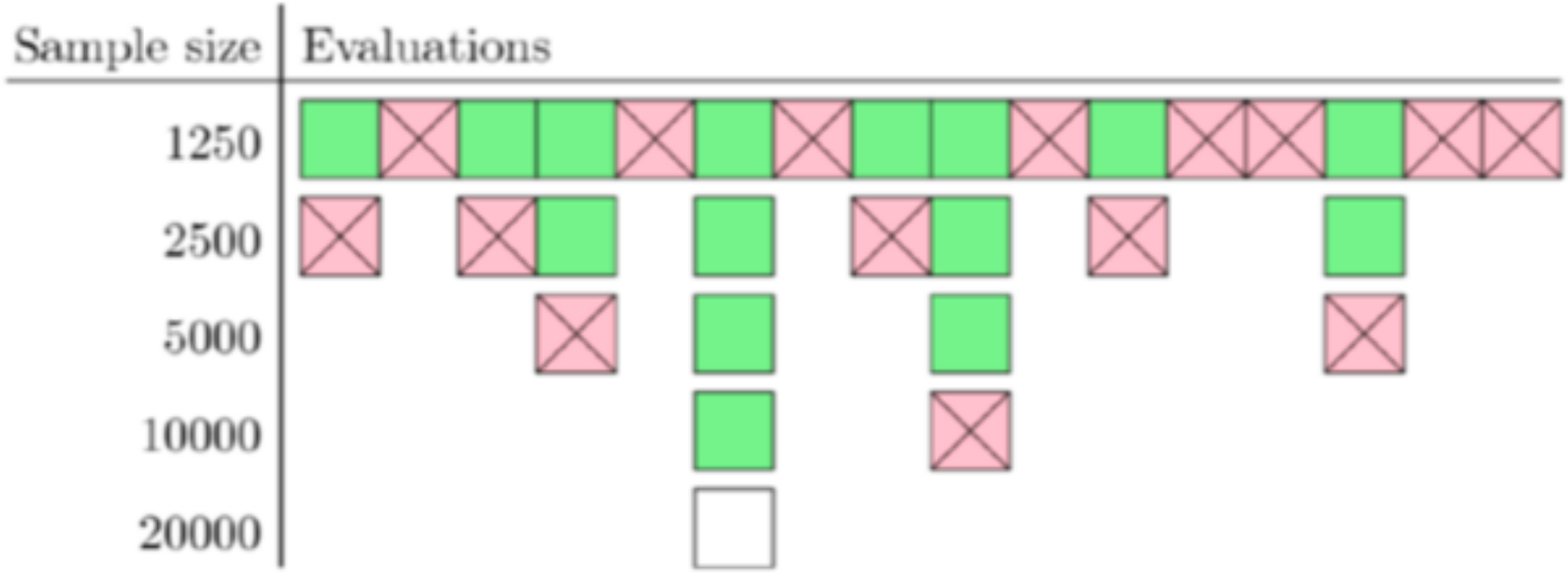
SPEEDING UP EVALUATIONS

Testing on a smaller sample of the data is faster, may give a good estimate of final performance (or maybe not)



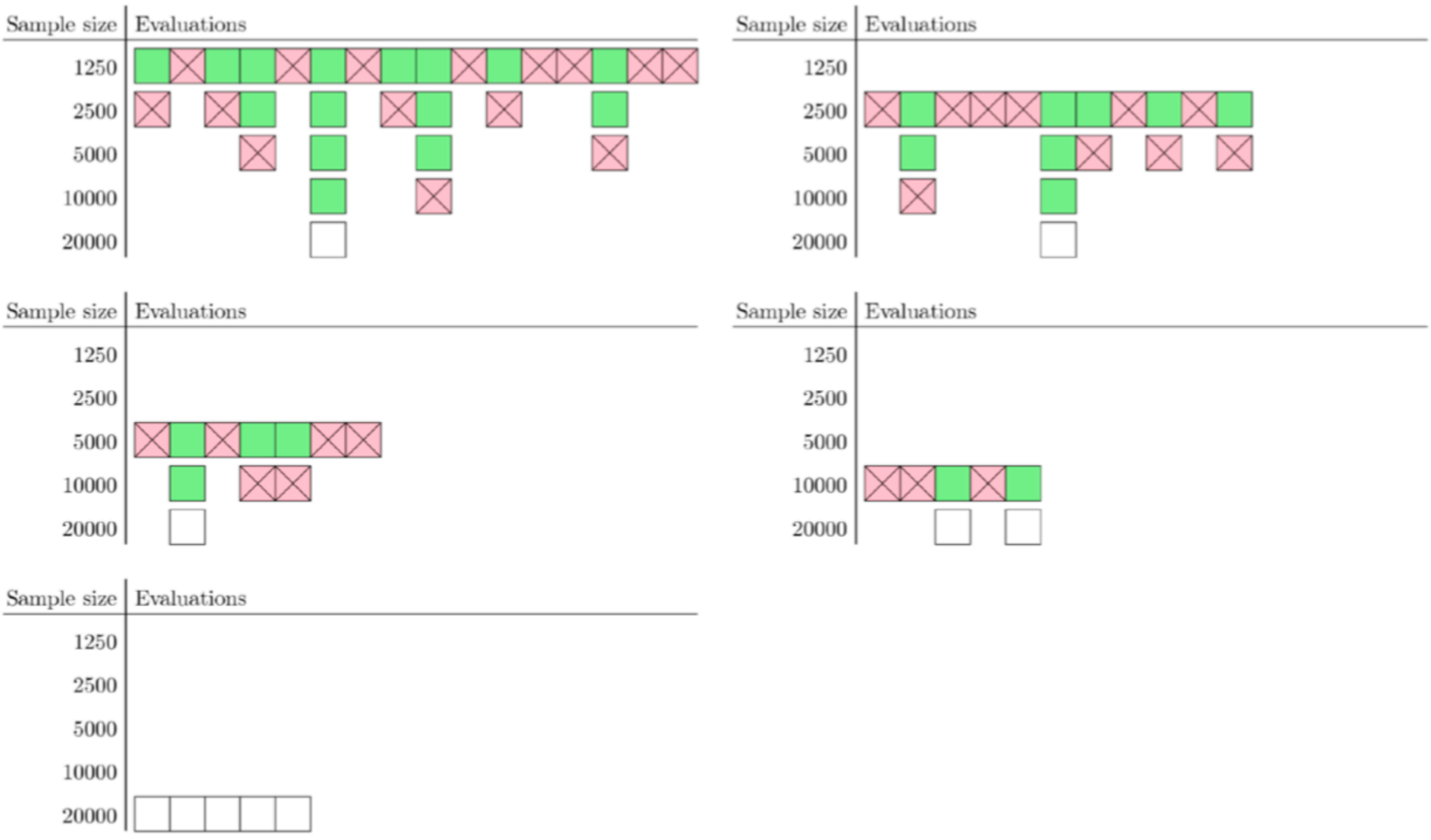
SPEEDING UP EVALUATIONS

Hyperband: based on ‘successive halving’



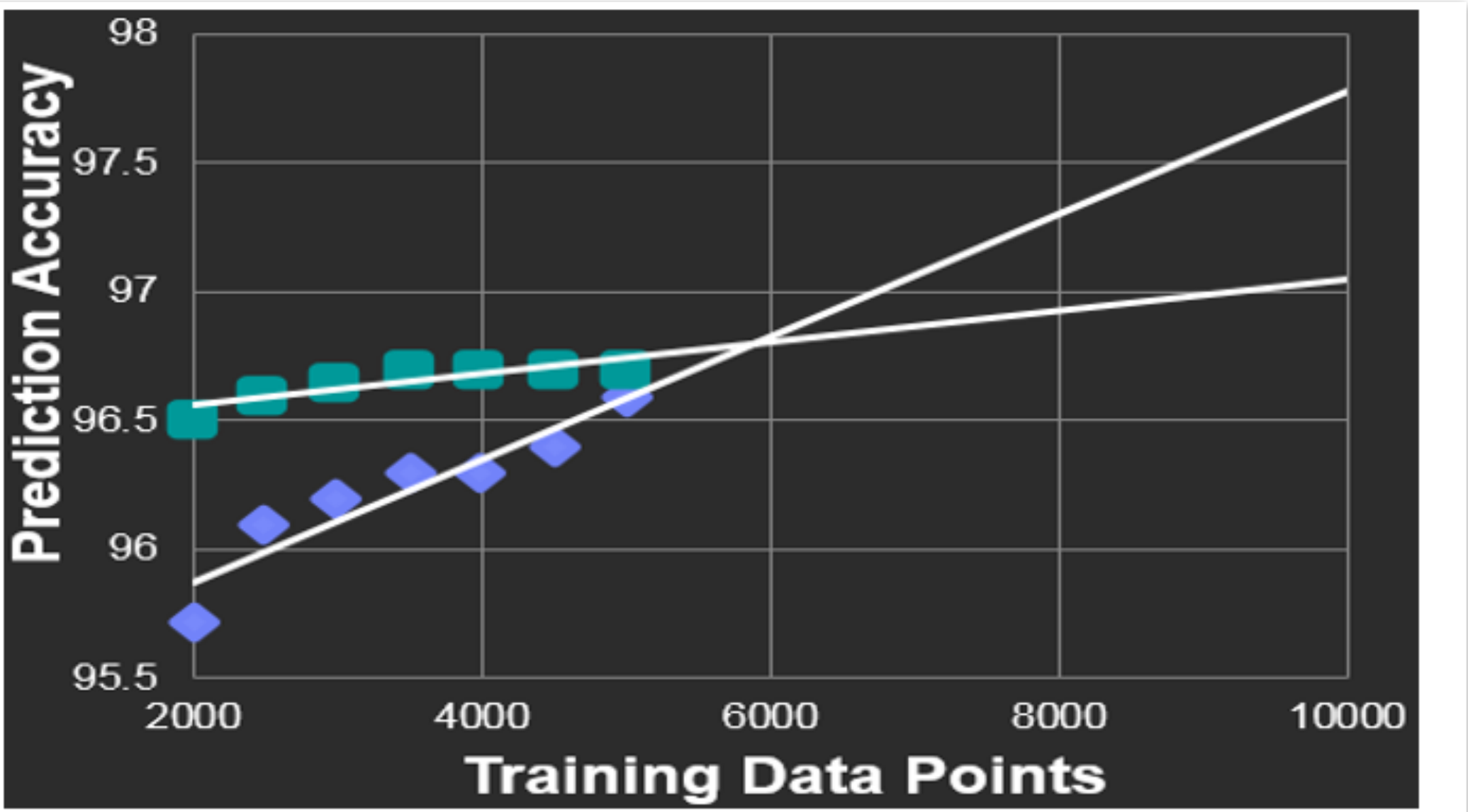
SPEEDING UP EVALUATIONS

Hyperband: based on ‘successive halving’



SPEEDING UP EVALUATIONS

DAUB: computes upper bounds



Learner Selection via Data Allocation using Upper Bounds (DAUB)

META-LEARNING

- Learn across datasets
  - **Warm-starting**: start with configurations/pipelines/ architectures that worked well on similar datasets
  - **Meta-models**:
    - Describe datasets with characteristics (metafeatures)
    - Predict performance, runtime,… for new datasets
- **Active testing**:
  - Don’t use meta-features, consider datasets to be similar if algorithms behave similarly
  - Learn dataset similarity and try promising configurations at the same time
- **Transfer learning**: reuse surrogate models from similar datasets

META-LEARNING + OPENML

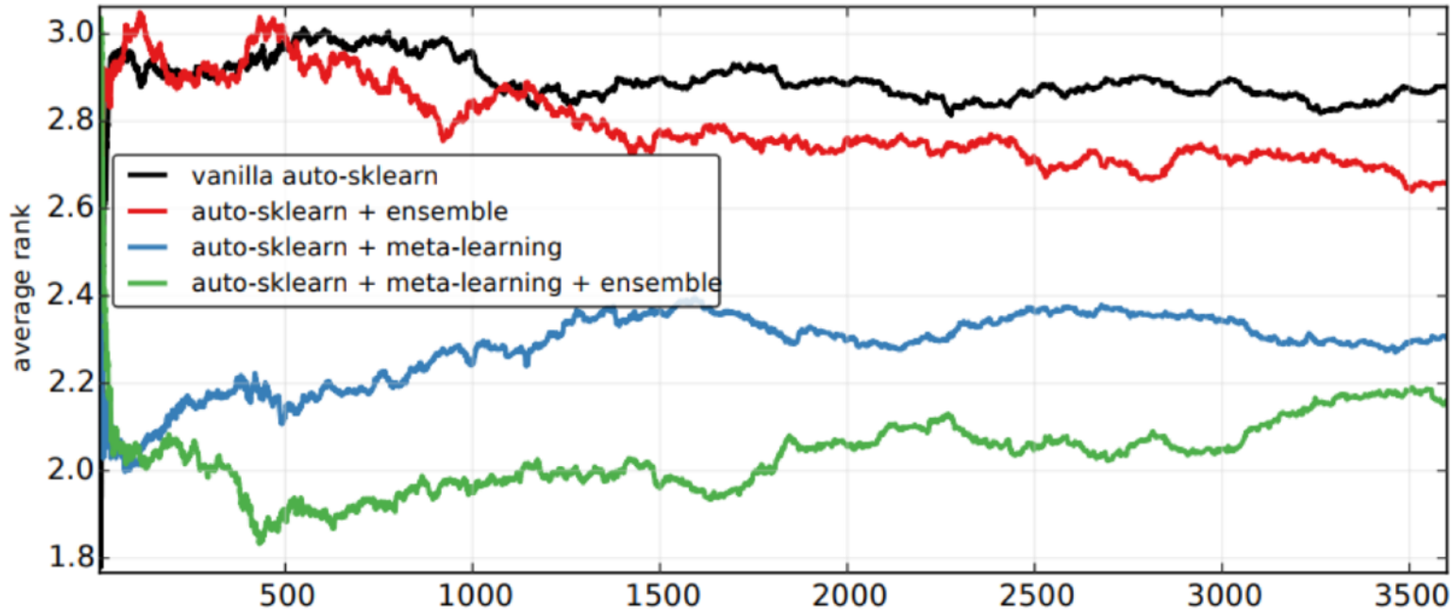
- **Find similar datasets**
  - 20,000+ datasets, each with 130+ meta-features
- **Reuse (millions of) prior model evaluations:**
  - Meta-models: E.g. predict performance or training time
- **Reuse results on many hyperparameter settings**
  - Surrogate models: predict best hyperparameter settings
  - Model the effect/importance of hyperparameters
- **Deeper understanding of algorithm performance:**
  - WHY models perform well (or not)
  - How is performance related to dataset properties
- **Never-ending AutoML**:
  - AutoML methods built on top of OpenML get increasingly better as more meta-data is added

META-FEATURES

- **Simple:** Number/log of instances, classes, features, dimensionality, NAs, …
- **Statistical:** Default acc, Min/max/mean distinct values, skewness, kurtosis, stdev, mutual information, …
- **Information-theoretic:** Entropy (class, num/nom feats), signal-noise ratio,…
- **Landmarkers** (AUC, ErrRate, Kappa): Decision stumps, Naive Bayes, kNN, RandomTree, J48(pruned), JRip, NBTree, lin.SVM, …
- **Subsampling landmarkers** (partial learning curves)
- **Streaming landmarkers:** Evaluations in previous window, previous best, per-algorithm significant wins/lossed
- **Change detectors:** HoeffdingAdwin, HoeffdingDDM

META-LEARNING: WARM STARTING

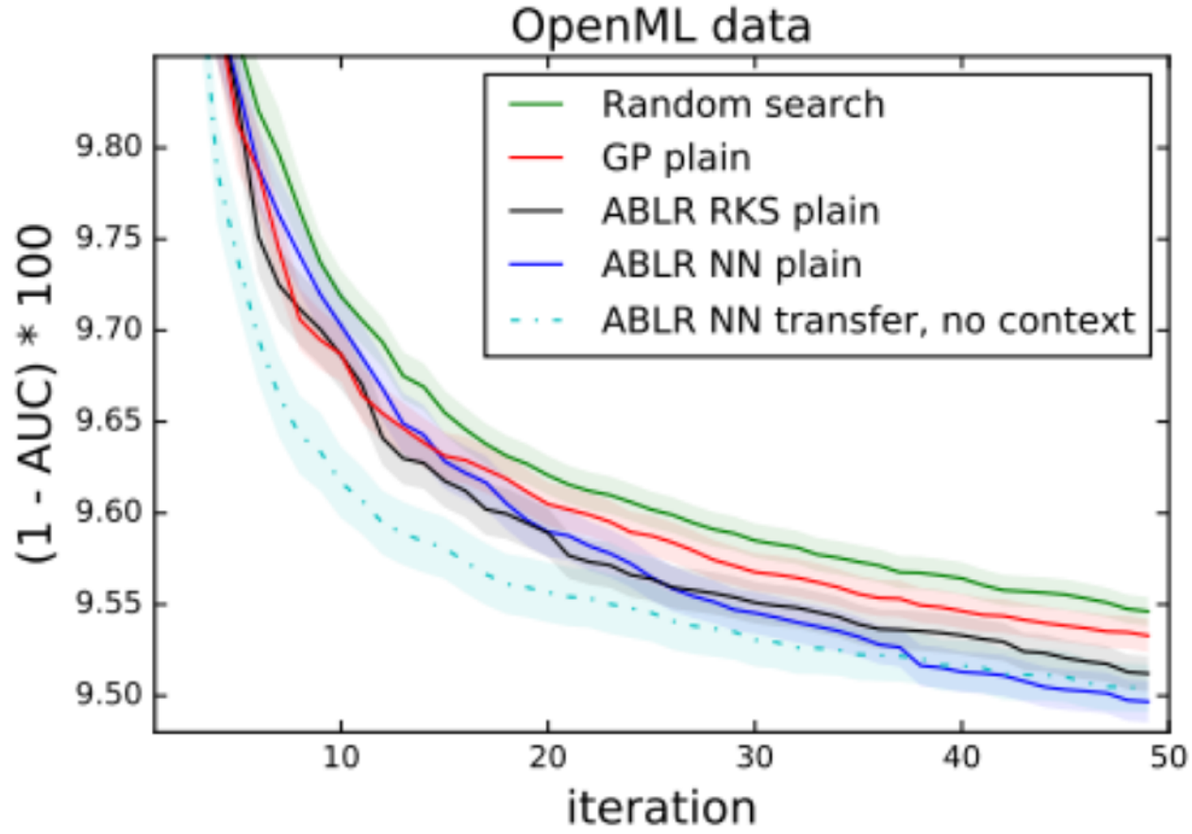
Auto-SKLearn: find similar datasets on OpenML, start with best configurations:



Matthias Feurer et al. (2016) NIPS

Adaptive Bayesian Lin. Regress.:

- build surrogate models from prior data on many OpenML tasks
- coupled through a neural net trained on evaluations on new dataset

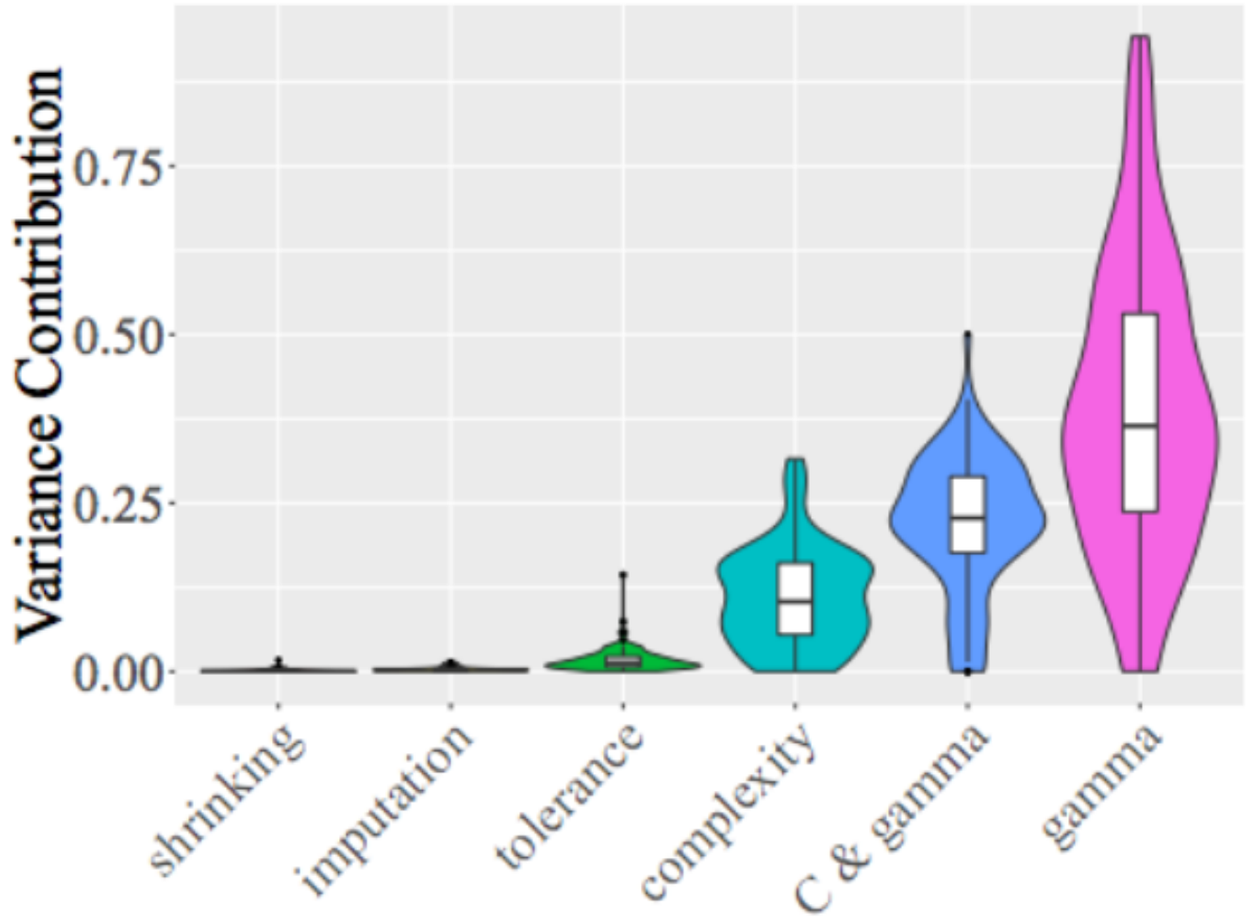


Valerio Perrone et al. (2017) MetaLearning@NIPS

META-LEARNING HYPERPARAMETER IMPORTANCE

SVM hyperparameter importance according to OpenML

Jan van Rijn et al. (2017) AutoML@ICML



Many other possible ways:

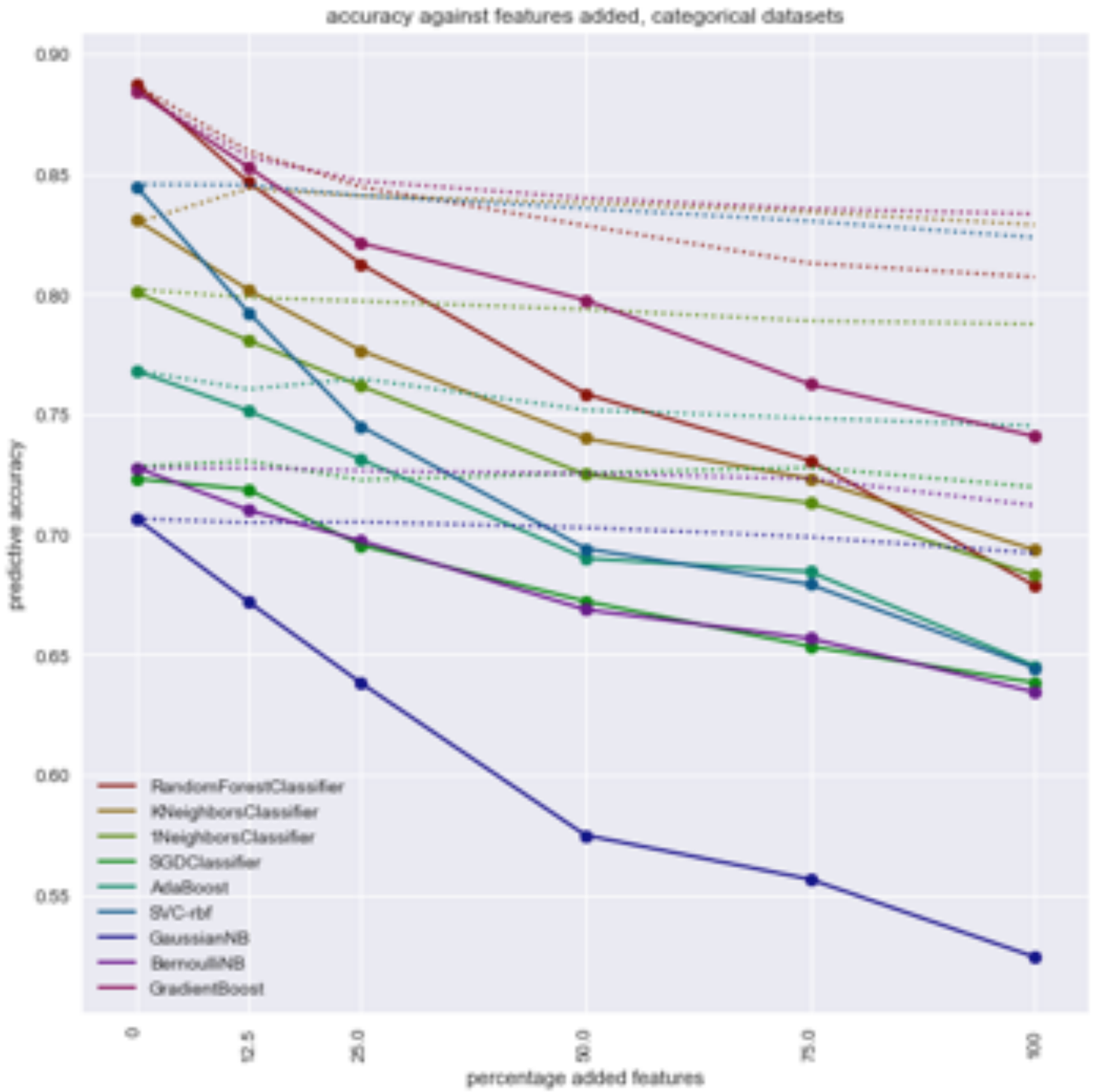
- Learn which hyperparameters should be tuned at all
- Learn good value ranges
- Learn good default values

META-LEARNING: PROFILING

Track score while adding:

- irrelevant features (dotted)
- redundant features (full)

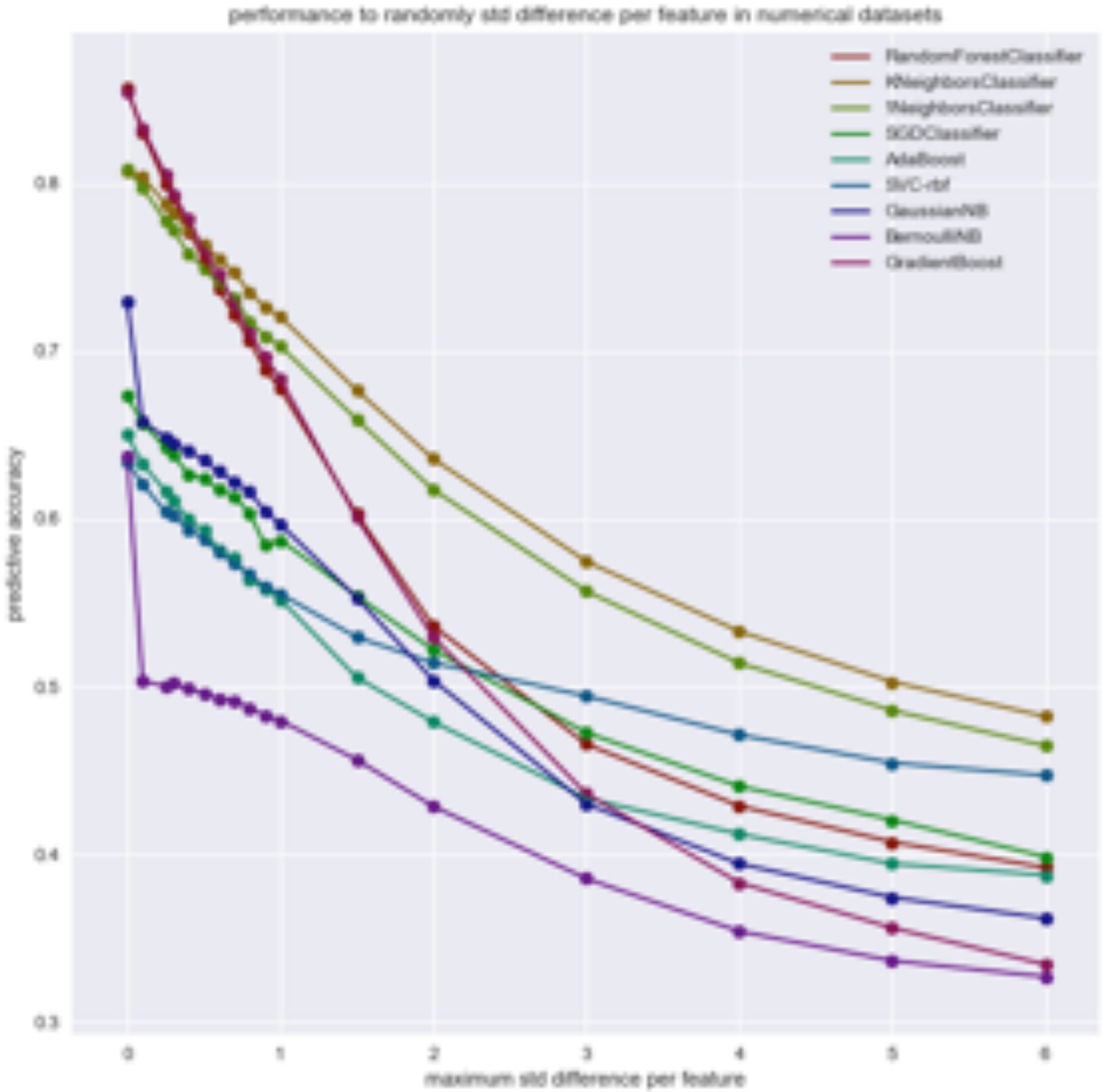
- Redundant features are bad (colinearity): SVMs, NBayes
- RandomForests, Boosting very sensitive to both!
- Tuning hyperparameters doesn't help -> select feats.



META-LEARNING: PROFILING

Add white noise to features.

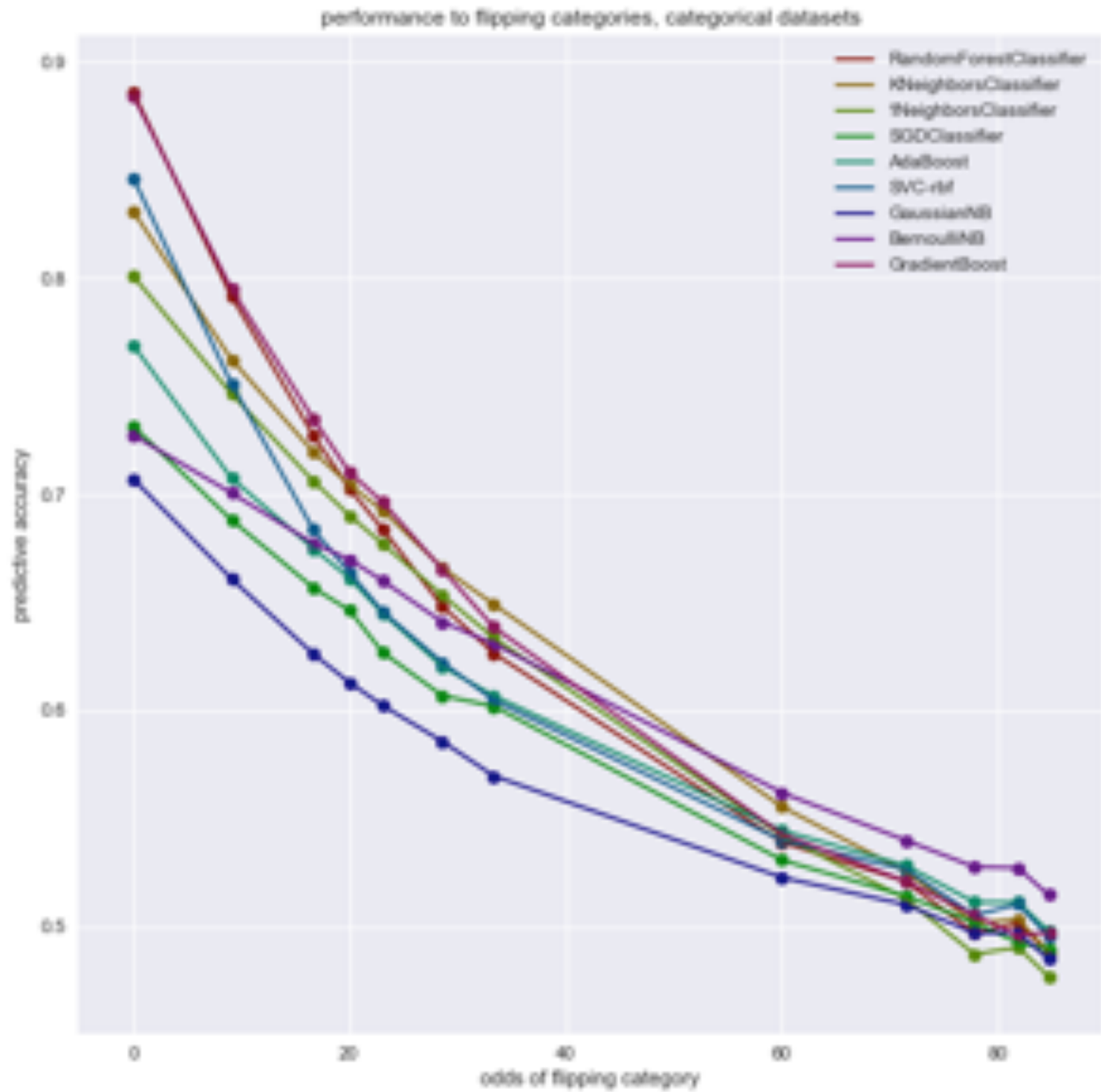
- RandomForests, Boosting, NBayes very sensitive
- kNN, SVM much less
- Hyperparameter tuning helps (regularization)!



META-LEARNING: PROFILING

Add noise to labels

- All algorithms hurt
- NBayes, kNN slightly more robust
- Tuning doesn't help.



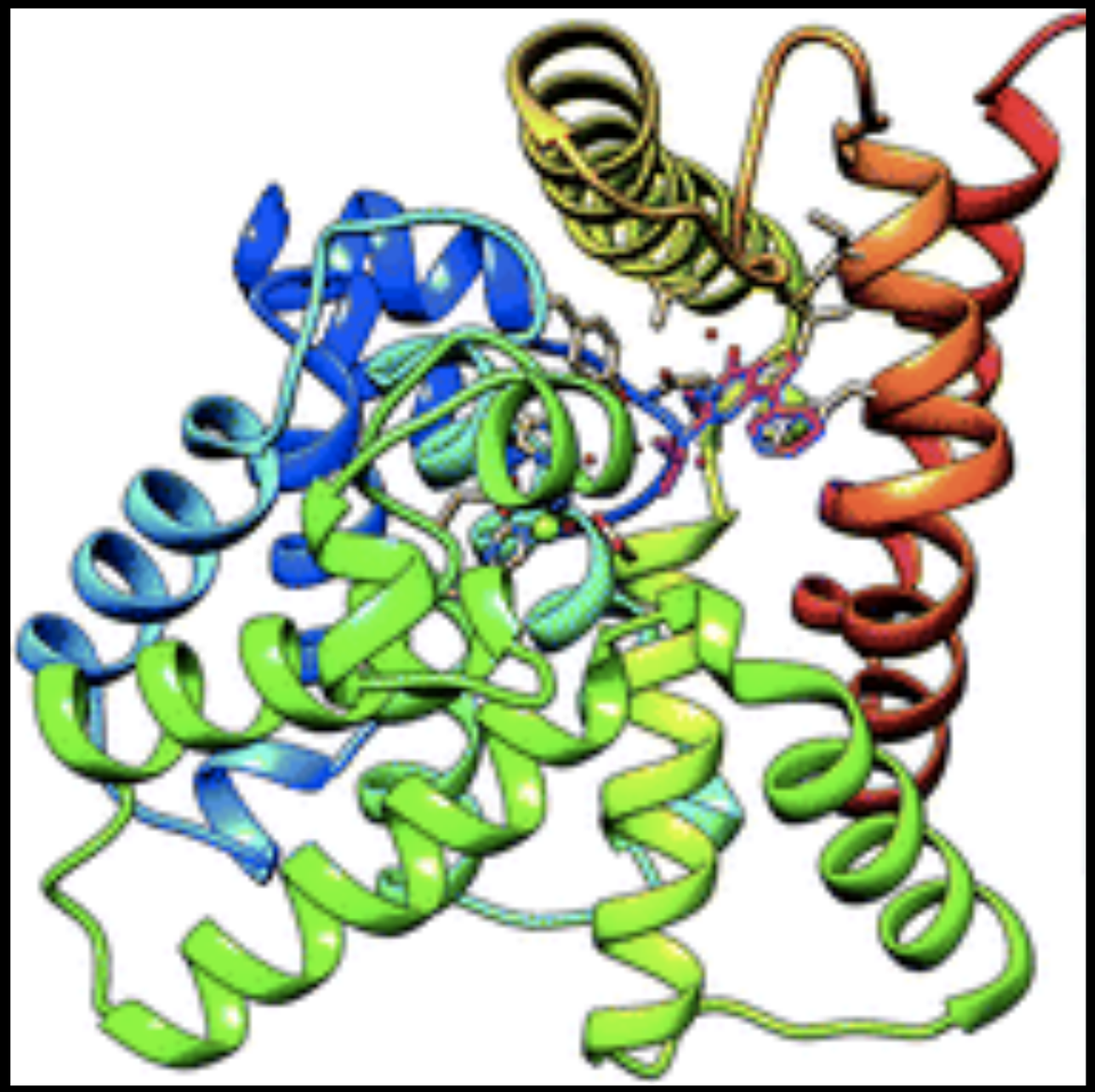
META-LEARNING: RUNTIME PREDICTION

- 100+ dataset meta-features and 100000+ experiment runtimes
- Build meta-models (Random Forest works well)
- Some algorithms more predictable than others

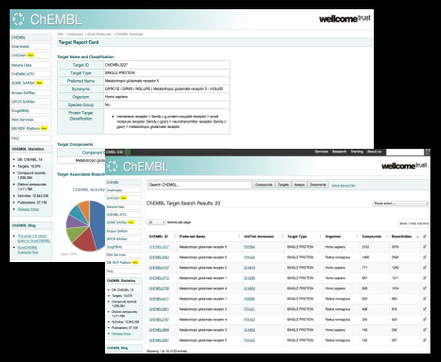
Regressor	RF	SVC	Tree	NB	Boosting	LR	kNN
Median	0.39	0.88	0.74	0.83	0.86	0.57	0.72
Mean	0.4	0.88	0.75	0.86	0.86	0.57	0.73
Extrapolate	0.44	0.80	0.45	1.33	0.33	0.84	1.09
RR	0.12	0.15	0.43	0.54	0.1	0.21	0.3
RF	**0.07**	**0.12**	**0.28**	**0.47**	**0.07**	**0.11**	**0.16**
SVR	0.39	0.87	0.72	0.86	0.84	0.56	0.71

Table 3: Mean Absolute Deviation of $\log_{10}$ runtime predictions. Best results are bold.

REAL-WORLD EXAMPLE: DRUG DISCOVERY



- Predict how well a drug inhibits protein function (and diseases depending on it)
- Problem: lab work (data) is **expensive**
- Solution:
  - use all available data on all available protein targets
  - use ***meta-learning*** to learn how to build optimal models for specific proteins



ChEMBL DB: 1.4M compounds,
10k proteins,12.8M activities

Molecule
representations

MW	LogP	TPSA	b1	b2	b3	b4	b5	b6	b7	b8	b9	
377.435	3.883	77.85	1	1	0	0	0	0	0	0	0	
341.361	3.411	74.73	1	1	0	1	0	0	0	0	0	...
197.188	-2.089	103.78	1	1	0	1	0	0	0	1	0	
346.813	4.705	50.70	1	0	0	1	0	0	0	0	0	
							.					
							:					

16,000 regression datasets
x52 pipelines (on OpenML)

all data on new protein →  → **optimal models to predict activity**

meta-model

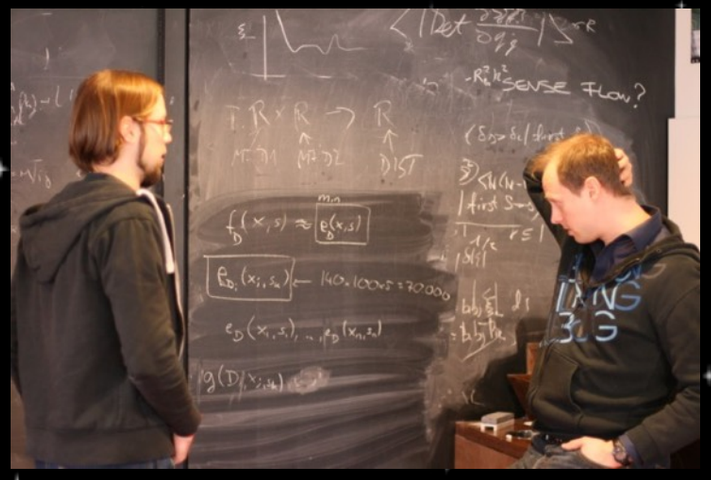
- **Auto-builds significantly better pipelines/models**
  (Olier et al., Machine Learning 107(1), 2018)
- **When data is scarce, build better models by transfer** from genetically similar targets (Sadawi et al, ChemInformatics, under review)
- Use trained models to build **better molecule representations**
- Collaboration with Exscientia Ltd (partner of GlaxoSmithKline)

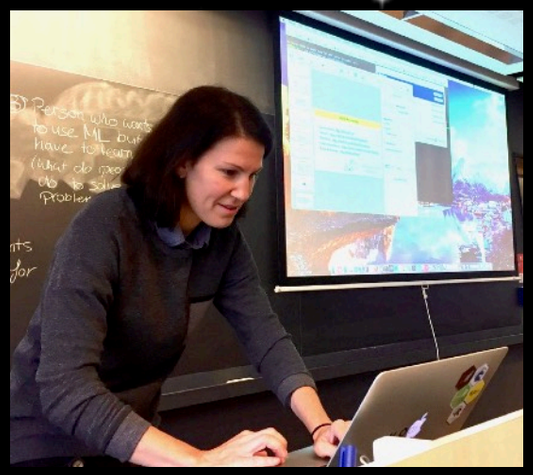
@open_ml

OpenML

Thank You

Questions?





ACKNOWLEDGEMENTS AND LINKS

- SSDS image data: http://www.sdss.org/
- Hyperparameter search plots: generate them yourself with mlr-mbo
  - https://mlr-org.github.io/First_release_of_mlrMBO_the_toolbox_for_Bayesian_Black_Box_Optimization/
- Bayesian optimization plots: https://arxiv.org/abs/1012.2599
- Bayesian optimization using Random Forests: based on a Jupiter Notebook by Marius Lindauer
- Thanks to several of my great students for creating many other plots: Jan van Rijn, Pieter Gijsbers, Marijn Knippenberg, Lieuwe Stooker, Chin-Fang Lin