PyData
MUMBAI

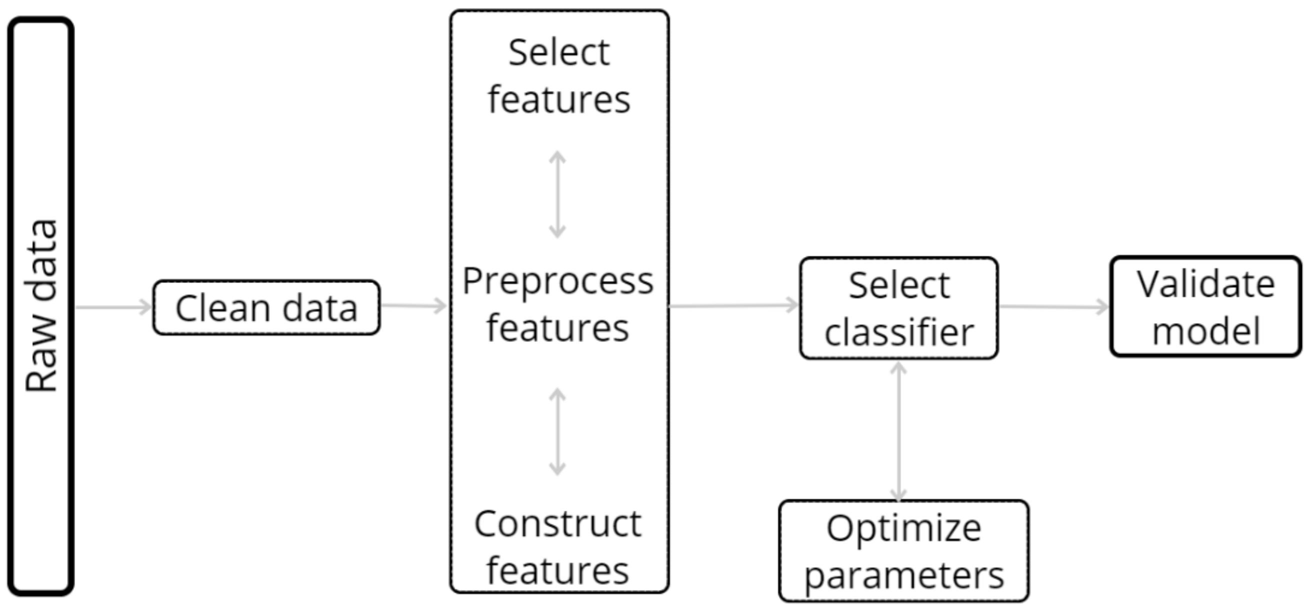
AutoML

- Pratik Bhavsar

Senior Data Scientist

Morningstar

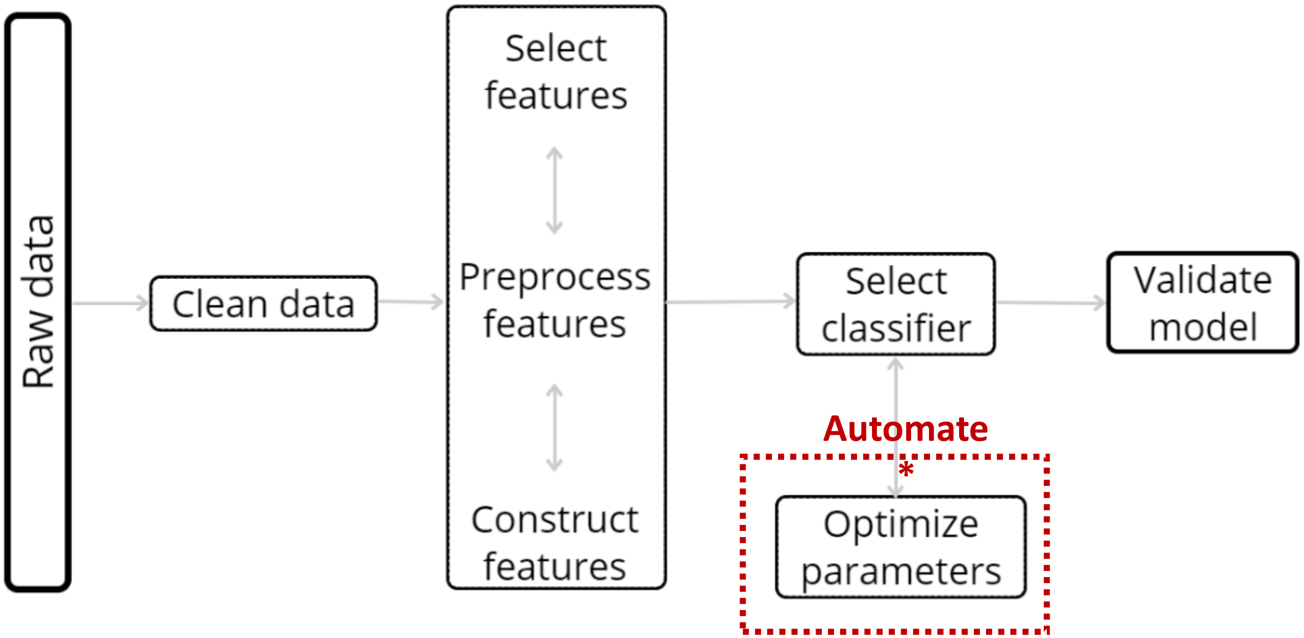
ML pipeline



Raw data
Clean data
Select features
Preprocess features
Construct features
Select classifier
Optimize parameters
Validate model

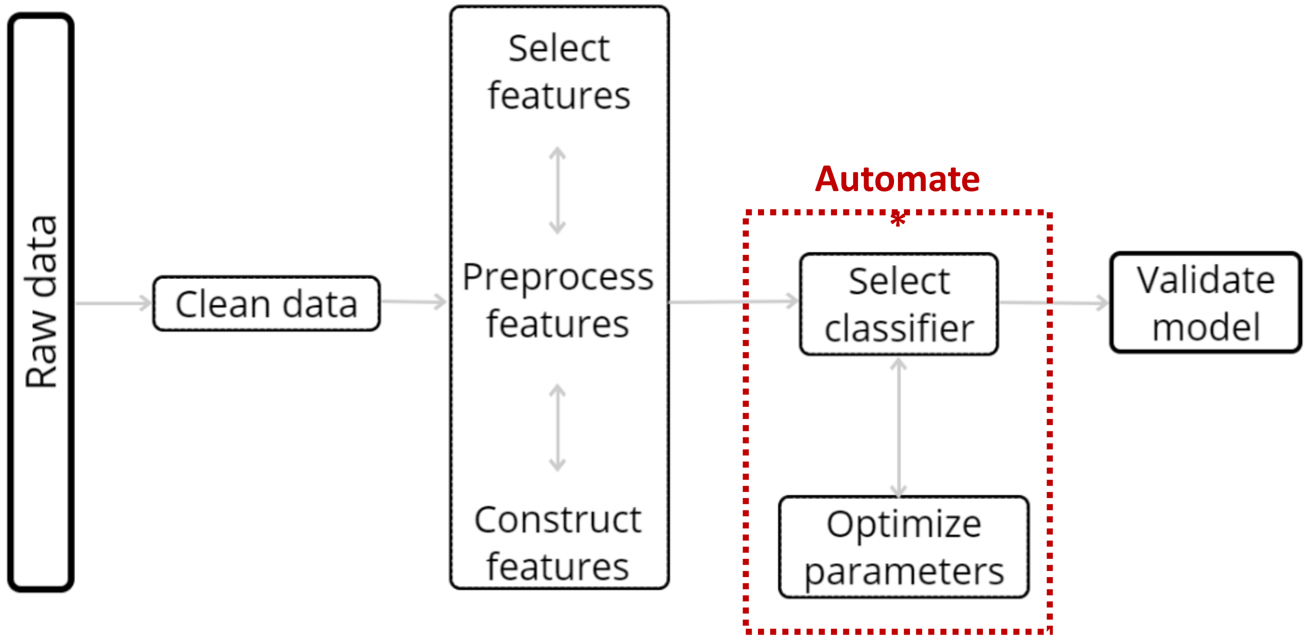
ML pipeline

*Hyperparameter Parameter Optimization (HPO)

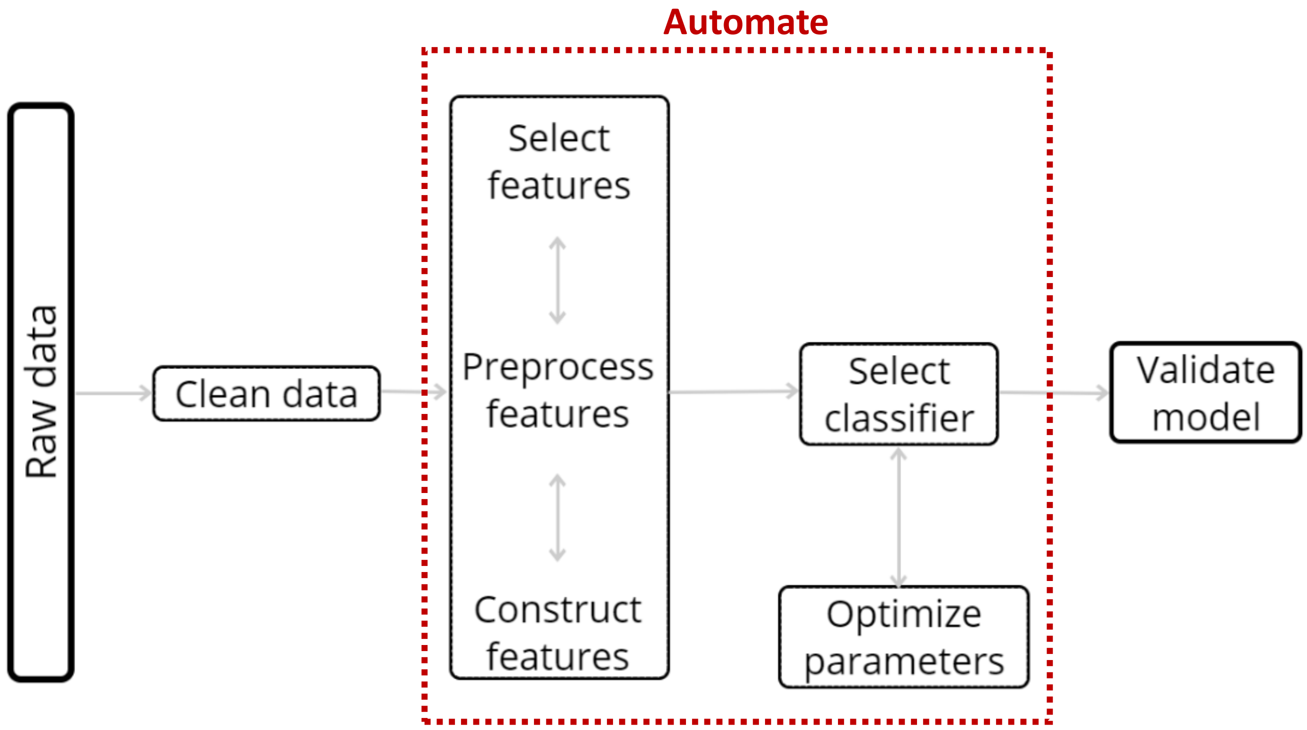


ML pipeline

*Combined Algorithm Selection and Hyperparameter optimization (CASH)



ML pipeline


Automate
Raw data
Clean data
Select features
Preprocess features
Construct features
Select classifier
Optimize parameters
Validate model

AutoML -> Pipeline Optimisation

- Data preprocessing
  - rescaling of the inputs
  - imputation of missing values
  - one-hot encoding*
  - balancing of the target classes
- Feature preprocessing methods
  - feature selection
  - kernel approximation
  - matrix decomposition
- Algorithms
  - Logistic
  - SVM
  - RandomForest
- Hyper-parameters

$$x^* = \underset{x \in X}{\operatorname{argmax}}\, P(x)$$

Hyper-parameters for RandomForest

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 2000]}
```

Search space - Scikit-learn parameters

name	$\#\lambda$	cat (cond)	cont (cond)
AdaBoost (AB)	4	1 (-)	3 (-)
Bernoulli naïve Bayes	2	1 (-)	1 (-)
decision tree (DT)	4	1 (-)	3 (-)
extreml. rand. trees	5	2 (-)	3 (-)
Gaussian naïve Bayes	-	-	-
gradient boosting (GB)	6	-	6 (-)
kNN	3	2 (-)	1 (-)
LDA	4	1 (-)	3 (1)
linear SVM	4	2 (-)	2 (-)
kernel SVM	7	2 (-)	5 (2)
multinomial naïve Bayes	2	1 (-)	1 (-)
passive aggressive	3	1 (-)	2 (-)
QDA	2	-	2 (-)
random forest (RF)	5	2 (-)	3 (-)
Linear Class. (SGD)	10	4 (-)	6 (3)

name	$\#\lambda$	cat (cond)	cont (cond)
extreml. rand. trees prepr.	5	2 (-)	3 (-)
fast ICA	4	3 (-)	1 (1)
feature agglomeration	4	3 ()	1 (-)
kernel PCA	5	1 (-)	4 (3)
rand. kitchen sinks	2	-	2 (-)
linear SVM prepr.	3	1 (-)	2 (-)
no preprocessing	-	-	-
nystroem sampler	5	1 (-)	4 (3)
PCA	2	1 (-)	1 (-)
polynomial	3	2 (-)	1 (-)
random trees embed.	4	-	4 (-)
select percentile	2	1 (-)	1 (-)
select rates	3	2 (-)	1 (-)
one-hot encoding	2	1 (-)	1 (1)
imputation	1	1 (-)	-
balancing	1	1 (-)	-
rescaling	1	1 (-)	-

Human error = bad model

- Missed predicting potential fraud
- Wrong diagnosis/prognosis
- Incorrect Recommendations
- Missed Churn Signal
- Promotional email to a wrong customer

Need For AutoML?

- Technical advantage
  - Eliminating human errors
  - Auto train models in production
  - Better models
  - Quick baseline solution
    - Hackathons
    - New project
  - Understand which pipeline works in which case
- Business advantage
  - Lack of data scientist
  - Faster development
    - Get business edge buy deploying models faster

Types of AutoML(1/3)

- Features
  - Feature engineering ex. DFS
  - HPO(Hyper parameter optimisation) ex.hyperopt
  - End-to-end ex. tpot
- Optimisation Algorithm
  - Genetic algorithm ex. tpot
  - Bayesian optimisation ex. auto-sklearn
- Algorithm search
  - Classical algorithms ex. tpot
  - Neural architecture ex. auto-keras

Types of AutoML(2/3)

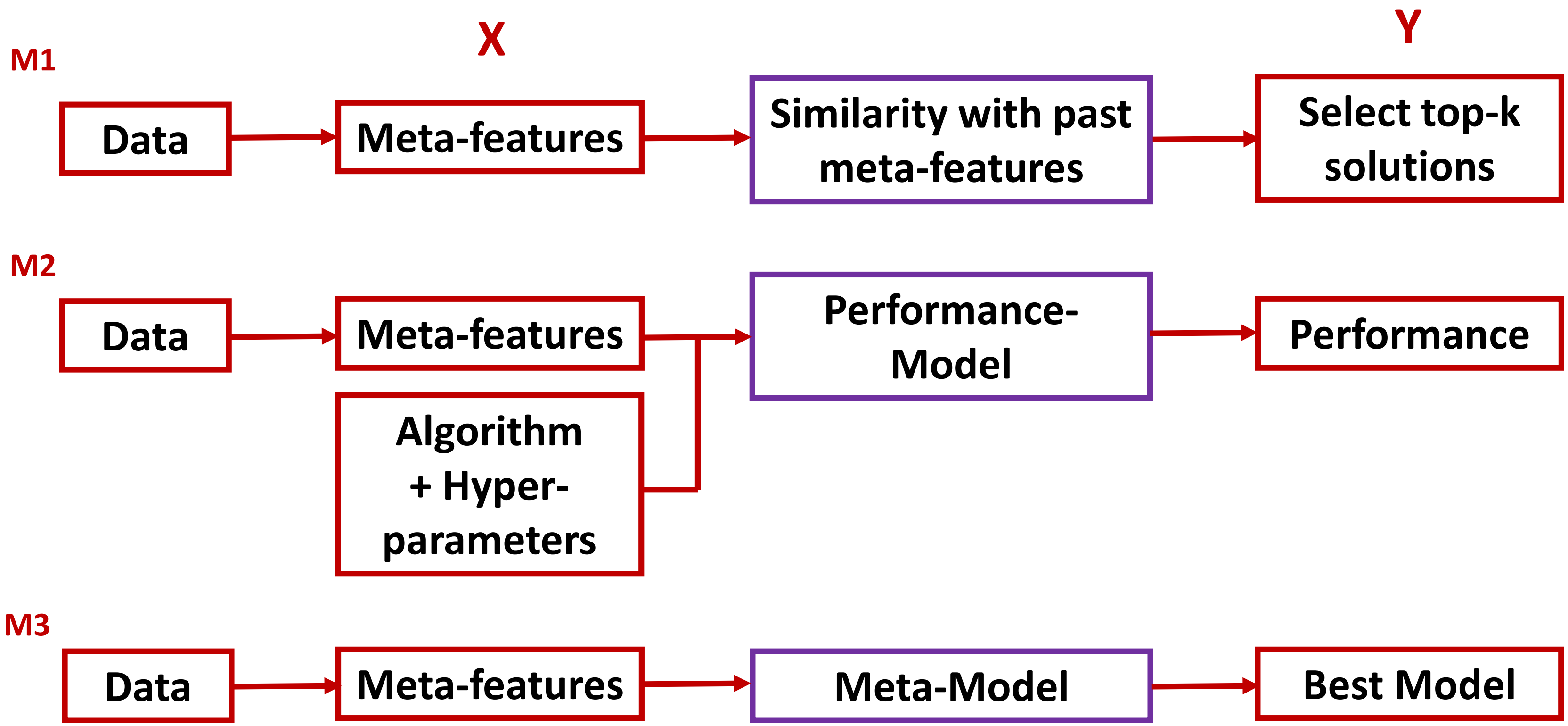
- Problem solvers
  - Regression
  - Classification
  - Time-series ex. h2o
  - NLP
    - Classical
    - Embedding
- Interface
  - Non-UI
  - UI (user-interface) ex. h20 driverless

Types of AutoML(3/3)

- Pricing
  - Free ex. tpot
  - Paid ex. All cloud and H2O driverless AI
- Language
  - Python ex. tpot, hyperopt, h2o
  - R ex. h2o
  - Java ex. weka
- Infrastructure integration
  - Library based ex. tpot, h2o, auto-sklearn
  - Cloud integrated ex. AWS Sagemaker, Microsoft Azure ML, Google AutoML

Meta-learning

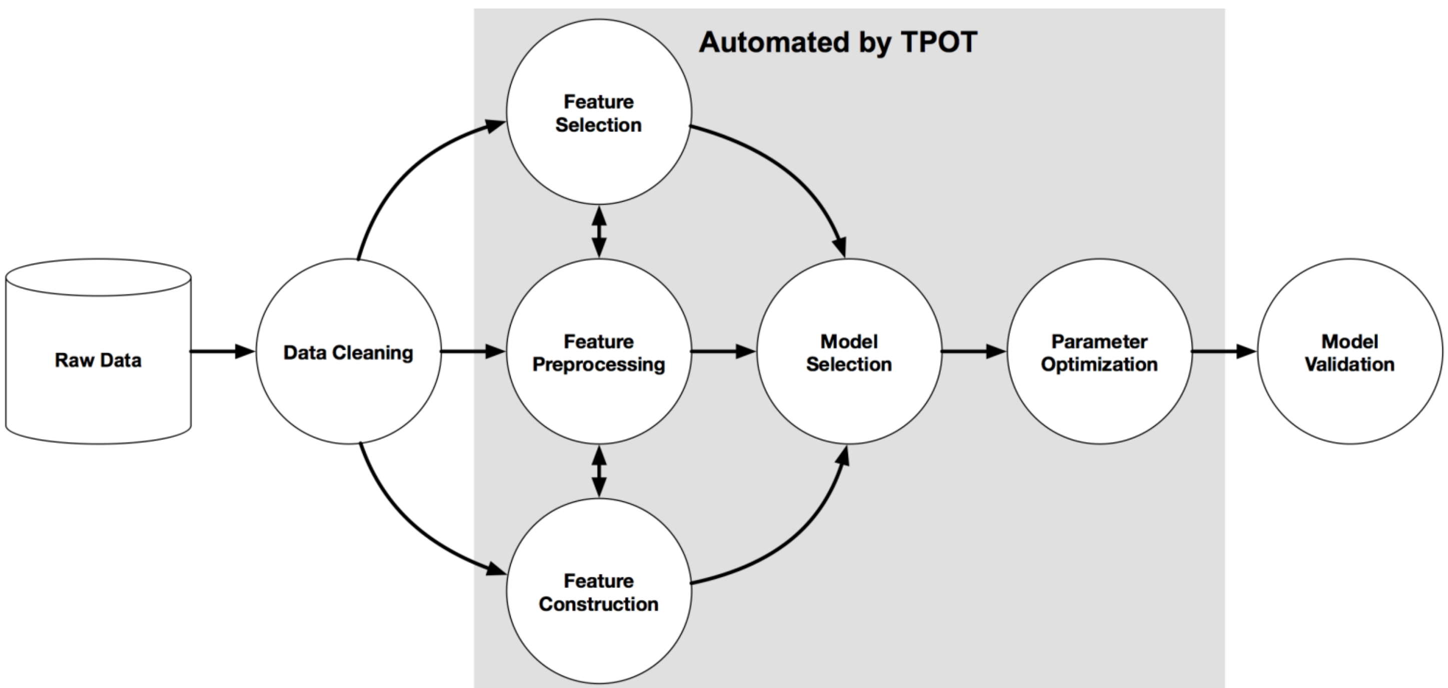
Learning to Learn



Task Meta-features

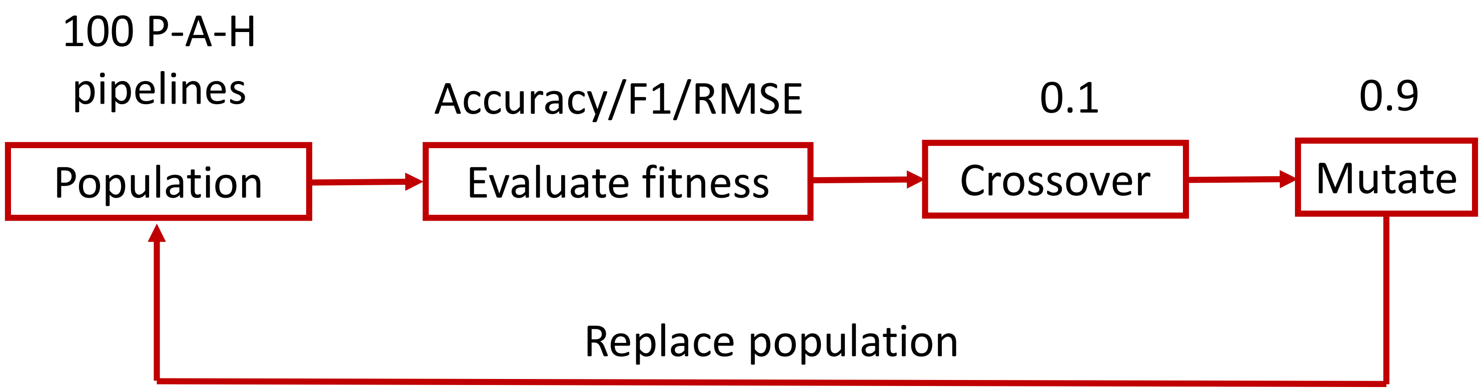
Name	Formula	Rationale	Variants
Nr instances	n	Speed, Scalability [99]	p/n, $log(n)$, log(n/p)
Nr features	p	Curse of dimensionality [99]	$log(p)$, % categorical
Nr classes	c	Complexity, imbalance [99]	ratio min/maj class
Nr missing values	m	Imputation effects [70]	% missing
Nr outliers	o	Data noisiness [140]	o/n
Skewness	$\frac{E(X-\mu_X)^3}{\sigma_X^3}$	Feature normality [99]	min,max,μ,σ,q_1, q_3
Kurtosis	$\frac{E(X-\mu_X)^4}{\sigma_X^4}$	Feature normality [99]	min,max,μ,σ,q_1, q_3
Correlation	$\rho_{X_1 X_2}$	Feature interdependence [99]	min,max,μ,σ,ρ_{XY} [157]
Covariance	$cov_{X_1 X_2}$	Feature interdependence [99]	min,max,μ,σ,cov_{XY}
Concentration	$\tau_{X_1 X_2}$	Feature interdependence [72]	min,max,μ,σ,τ_{XY}
Sparsity	sparsity(X)	Degree of discreteness [142]	min,max,μ,σ
Gravity	gravity(X)	Inter-class dispersion [5]	
ANOVA p-value	$p_{val_{X_1 X_2}}$	Feature redundancy [70]	$p_{val_{XY}}$ [157]
Coeff. of variation	$\frac{\sigma_Y}{\mu_Y}$	Variation in target [157]	
PCA ρ_{λ_1}	$\sqrt{\frac{\lambda_1}{1+\lambda_1}}$	Variance in first PC [99]	$\frac{\lambda_1}{\sum_i \lambda_i}$ [99]
PCA skewness		Skewness of first PC [48]	PCA kurtosis [48]
PCA 95%	$\frac{dim_{95\%var}}{p}$	Intrinsic dimensionality [9]	
Class probability	$P(\texttt{C})$	Class distribution [99]	min,max,μ,σ

tpot


Automated by TPOT
Raw Data
Data Cleaning
Feature Selection
Feature Preprocessing
Feature Construction
Model Selection
Parameter Optimization
Model Validation

Genetic algorithms

P – Pre-processing
A – Algorithm
H – Hyper-paramters



https://natureofcode.com/book/chapter-9-the-evolution-of-code/

tpot – pipeline

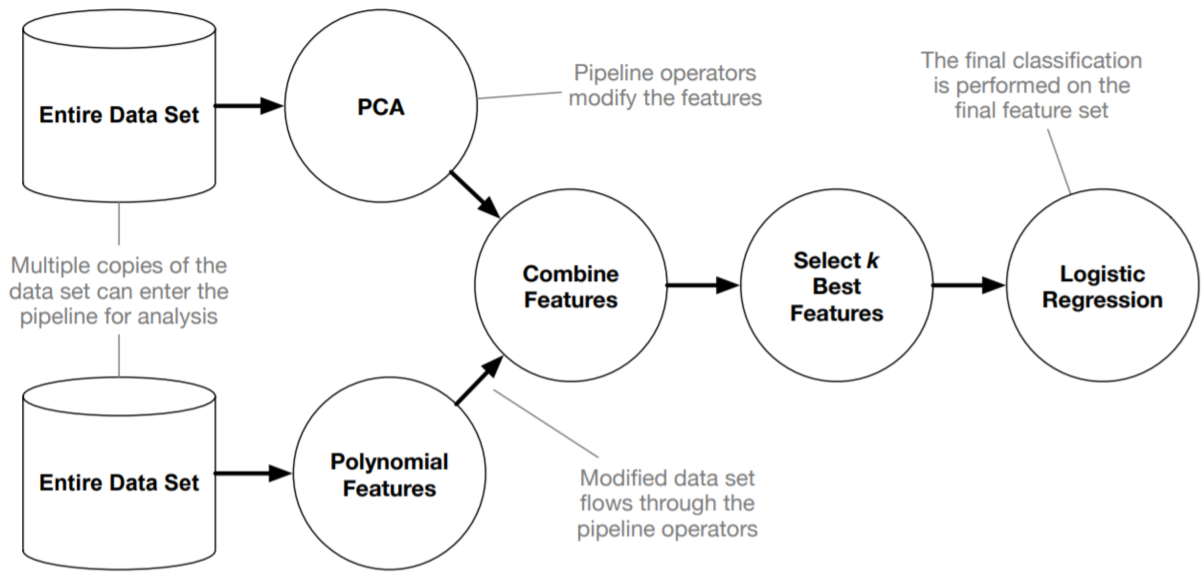


Figure 8.1: An example tree-based pipeline from TPOT. Each circle corresponds to a machine learning operator, and the arrows indicate the direction of the data flow.

tpot - parameters

- **generations** The default is 100.
- **population_size**: The default is 100.
- **offspring_size**: The default is 100.
- **mutation_rate**: Default is 0.9
- **crossover_rate**: Default is 0.1
- **cv**: Cross-validation strategy used when evaluating pipelines. The default is 5.

- **scoring:**
  accuracy, average_precision, roc_auc, recall, etc. The default is accuracy.
- **max_time_mins**
- **max_eval_time_min**

The default is 5.

- **early_stop**
- **n_jobs**
- **Subsample:** : Must be in the range (0.0, 1.0]. The default is 1.

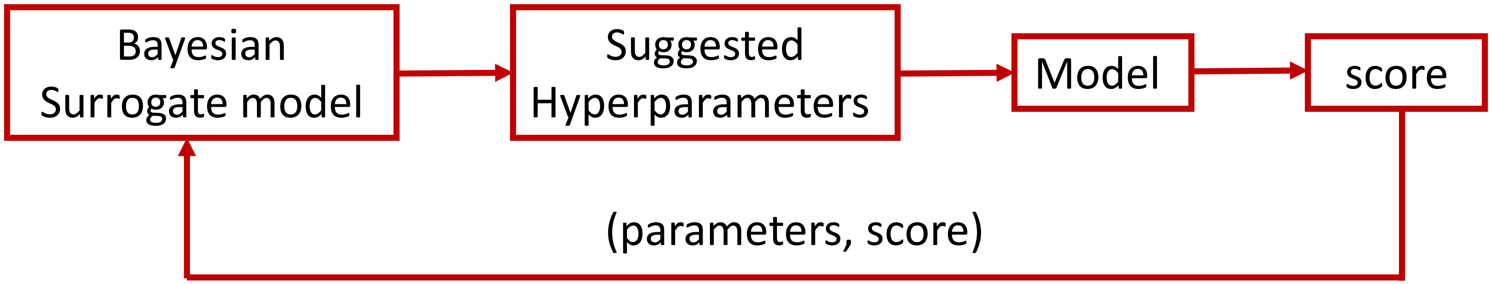
HPO

- Manual
- Grid search
- Random search
- Bayesian optimization

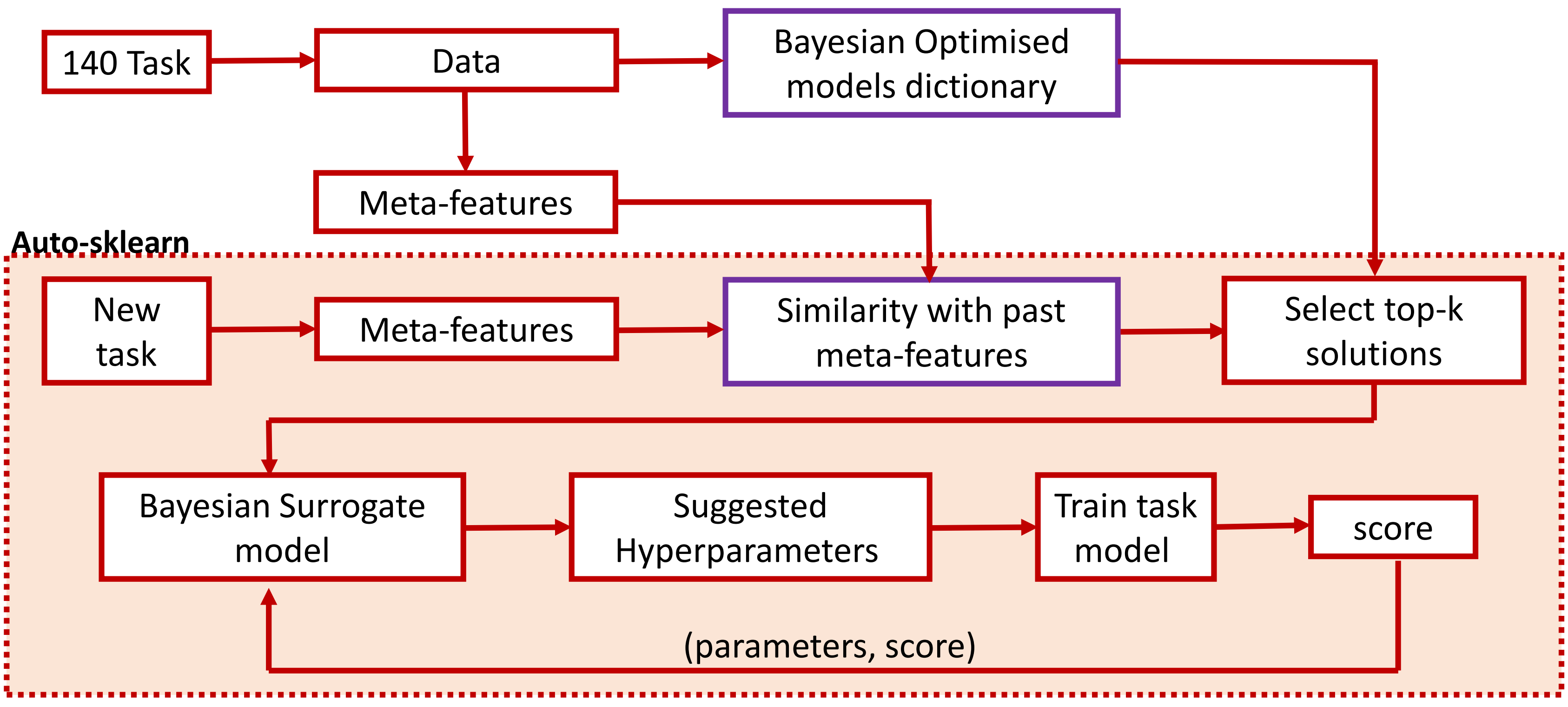
Hyper-parameters for RandomForest

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 100, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 2000]}
```

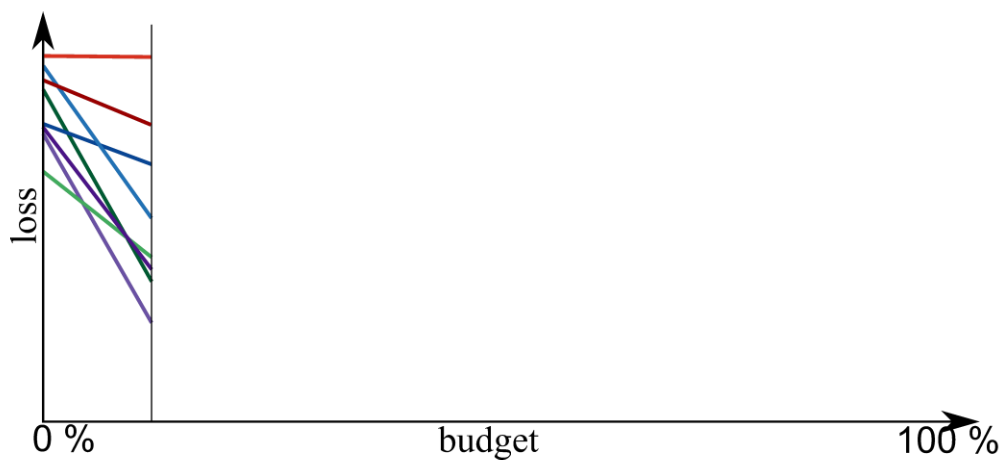
Bayesian Optimisation



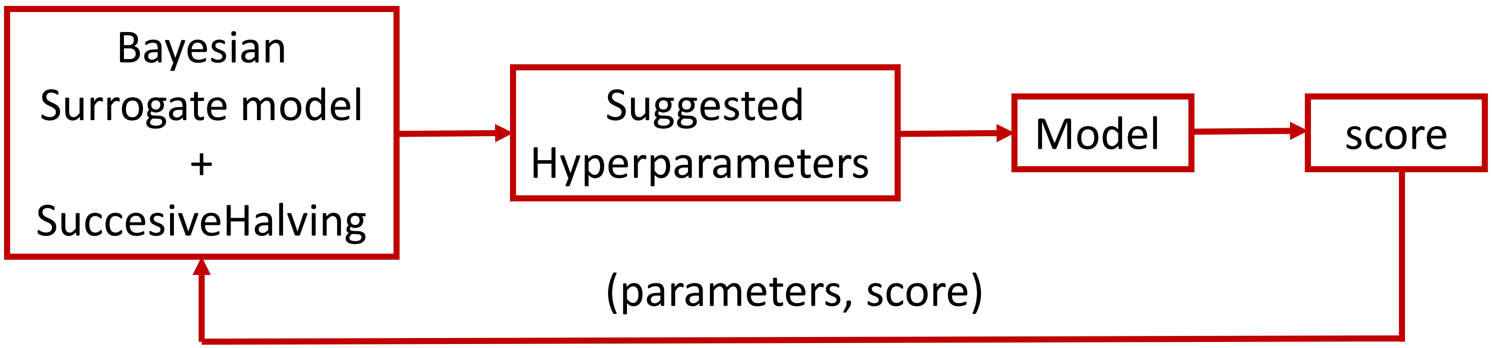
Auto-sklearn



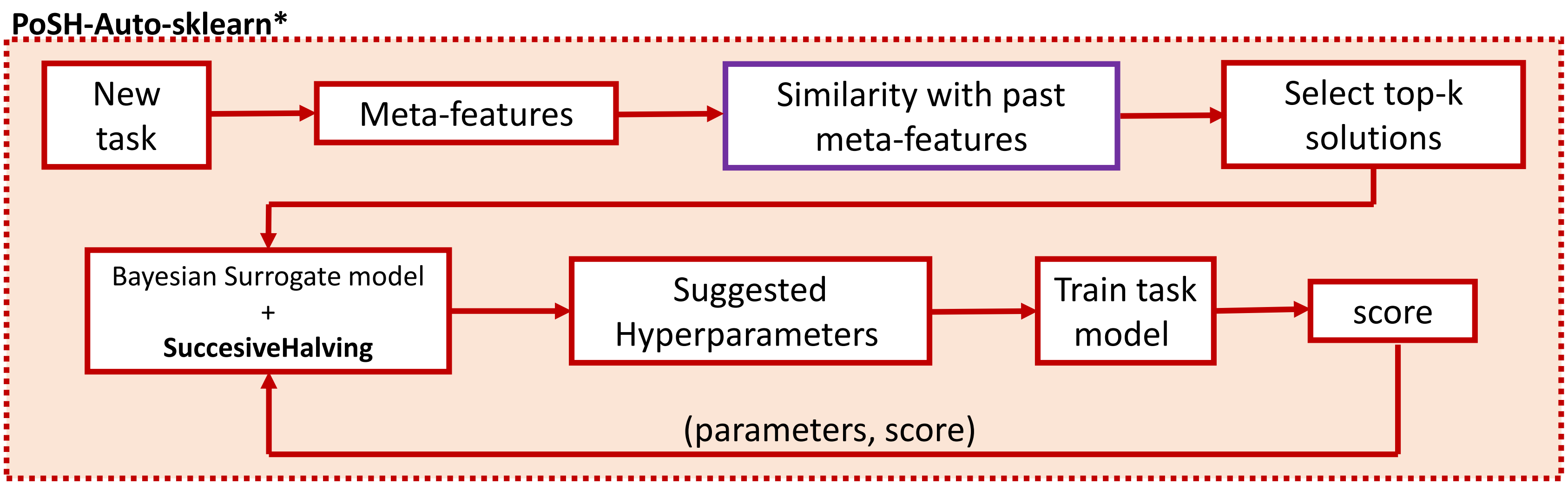
Hyperband - SuccesiveHalving



BOHB(Bayesian Optimisation – Hyperband)



AutoML challenge (2017-2018)



*ML Freiburg lab is the world champion in automatic machine learning (AutoML)

Auto-sklearn

- 110 conditional hyperparameters
  - 15 classification algorithms
  - 14 preprocessing methods
  - 4 data preprocessing methods
- Data preprocessing
  - rescaling of the inputs
  - imputation of missing values
  - one-hot encoding*
  - balancing of the target classes

14 feature preprocessing methods

- feature selection
- matrix decomposition
- embeddings
- polynomial feature expansion
- classifier for feature selection**

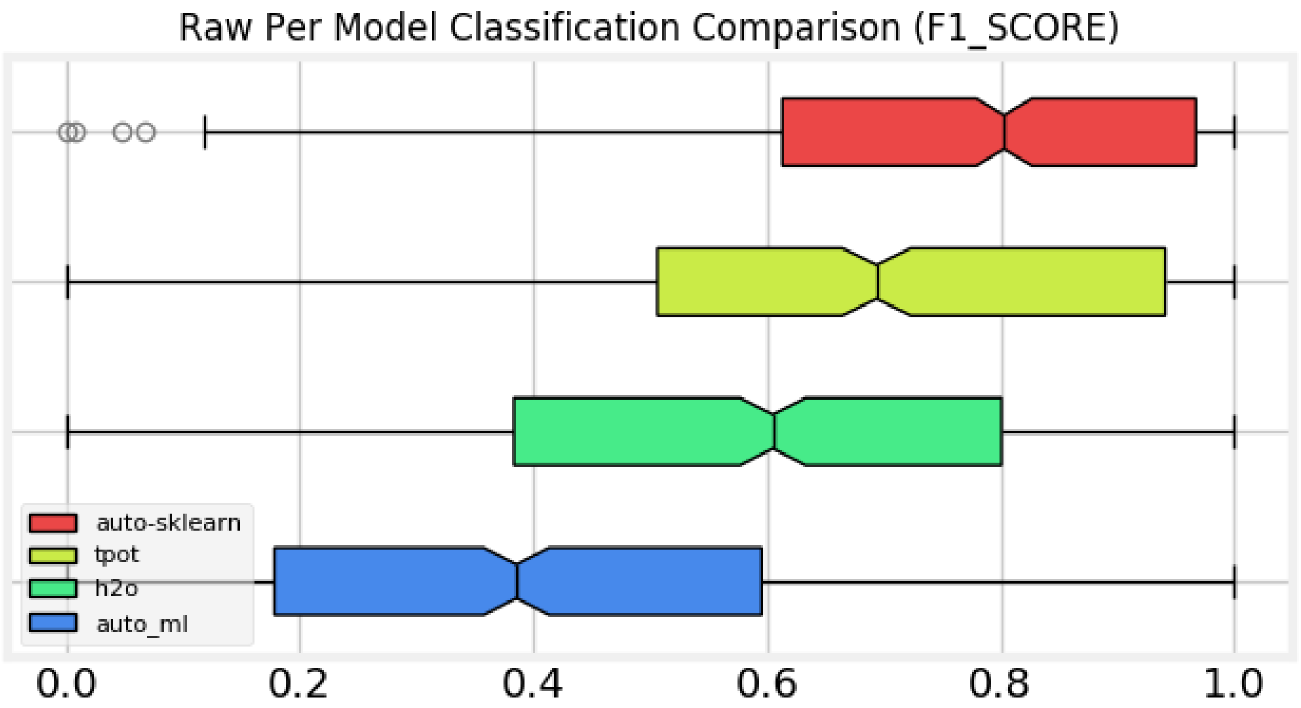
*Since scikit-learn methods are restricted to numerical input values, we always transformed data by applying a one-hot encoding to categorical features. In order to keep the number of dummy features low, we configured a percentage threshold and a value occurring more rarely than this percentage was transformed to a special other value.
** L1-regularized linear SVMs fitted to the data can be used for feature selection by eliminating features corresponding to zero-valued model coefficients.

Auto-sklearn

name	$\#\lambda$	cat (cond)	cont (cond)
AdaBoost (AB)	4	1 (-)	3 (-)
Bernoulli naïve Bayes	2	1 (-)	1 (-)
decision tree (DT)	4	1 (-)	3 (-)
extreml. rand. trees	5	2 (-)	3 (-)
Gaussian naïve Bayes	-	-	-
gradient boosting (GB)	6	-	6 (-)
kNN	3	2 (-)	1 (-)
LDA	4	1 (-)	3 (1)
linear SVM	4	2 (-)	2 (-)
kernel SVM	7	2 (-)	5 (2)
multinomial naïve Bayes	2	1 (-)	1 (-)
passive aggressive	3	1 (-)	2 (-)
QDA	2	-	2 (-)
random forest (RF)	5	2 (-)	3 (-)
Linear Class. (SGD)	10	4 (-)	6 (3)

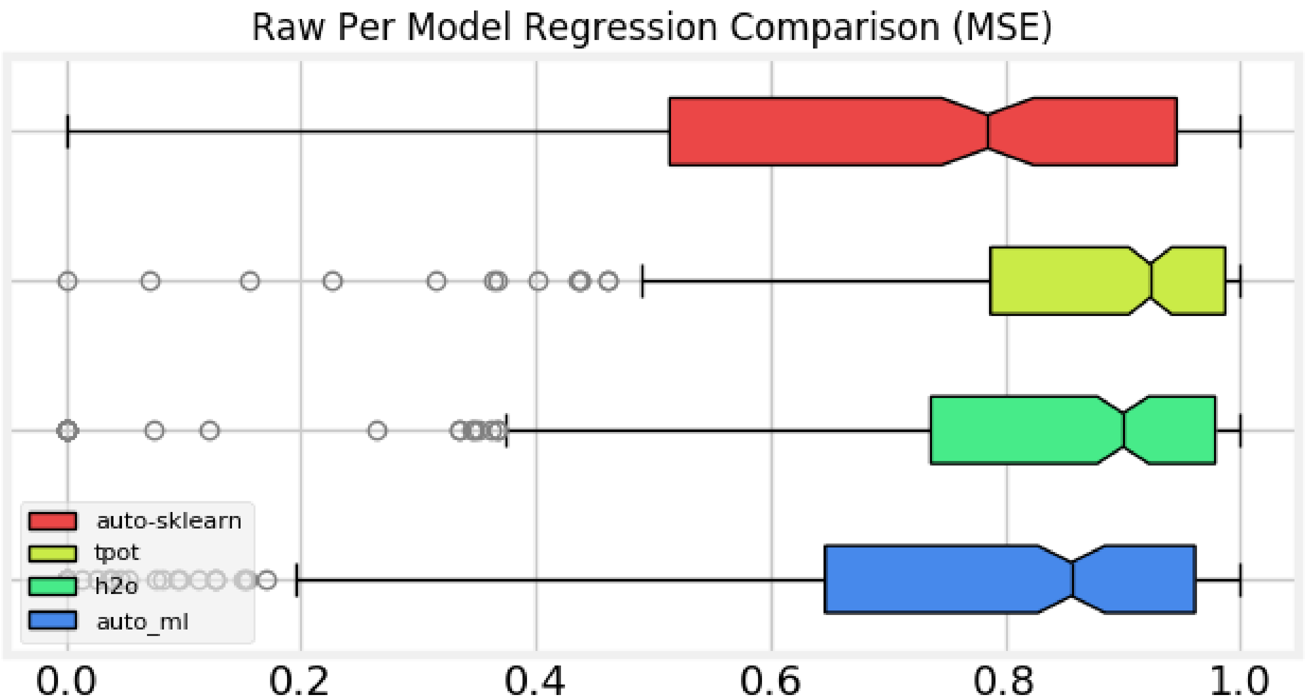
name	$\#\lambda$	cat (cond)	cont (cond)
extreml. rand. trees prepr.	5	2 (-)	3 (-)
fast ICA	4	3 (-)	1 (1)
feature agglomeration	4	3 ()	1 (-)
kernel PCA	5	1 (-)	4 (3)
rand. kitchen sinks	2	-	2 (-)
linear SVM prepr.	3	1 (-)	2 (-)
no preprocessing	-	-	-
nystroem sampler	5	1 (-)	4 (3)
PCA	2	1 (-)	1 (-)
polynomial	3	2 (-)	1 (-)
random trees embed.	4	-	4 (-)
select percentile	2	1 (-)	1 (-)
select rates	3	2 (-)	1 (-)
one-hot encoding	2	1 (-)	1 (1)
imputation	1	1 (-)	-
balancing	1	1 (-)	-
rescaling	1	1 (-)	-

Comparison – 57 Classification Tasks



https://arxiv.org/pdf/1808.06492.pdf

Comparison – 30 Regression Tasks



https://arxiv.org/pdf/1808.06492.pdf

AutoML for Neural Networks

- Neural Architecture Search(NAS)
- Efficient Neural Architecture Search
- Differentiable architecture search (D
- RL
    - AmoebaNet
    - Adanet

Papers so far...

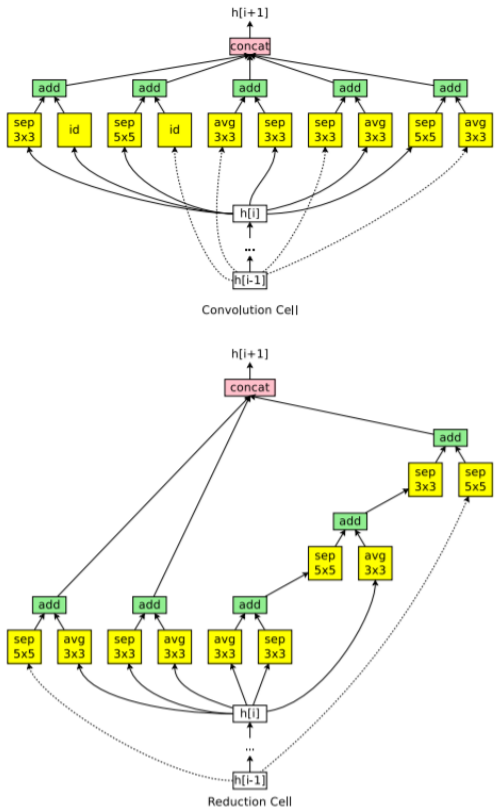


Figure 8. ENAS cells discovered in the micro search space.

AutoML Comparison

AUTO ML SOLUTIONS

As of May 2018

SOLUTIONS	OPEN SOURCE ?	# CONTRIBUTORS	# STARS	BACKEND FRAMEWORKS	OPTIMIZATION ALGORITHM	OTHER FEATURES
Amazon AWS Machine learning	No	-	-	TensorFlow, PyTorch...	-	cloud, access to GPUs, can be combined with other tools (data storage, querying...)
Autosklearn	Yes	25	2,218	Sklearn	bayesian	-
AutoWeka	Yes	4	124	Java WEKA ML library	bayesian grid search	-
DataRobot	No	-	-	-	-	handles deployment, cloud
Google Cloud HyperTune	No	-	-	Tensorflow	bayesian	cloud, data exploration and preparation through Google Cloud Datalab
H2O	Yes	103	3,075	H2O	grid search	can be distributed
H2O Driverless	No	-	-	H2O	-	performs features engineering
Hyperopt	Yes	25	2,078	-	random tree parzen estimator	-
IBM Watson	No	-	-	-	-	-
MLBox	Yes	3	482	Sklearn, Keras, XGBoost...	tree parzen estimator	basic data pre-processing, feature selection
PyBrain	Yes	32	2,544	homemade + libraries (LIBSVM)	metaheuristics grid search	-
TPOT	Yes	32	4,002	sklearn	genetic	-

https://hackernoon.com/a-brief-overview-of-automatic-machine-learning-solutions-automl-2826c7807a2a

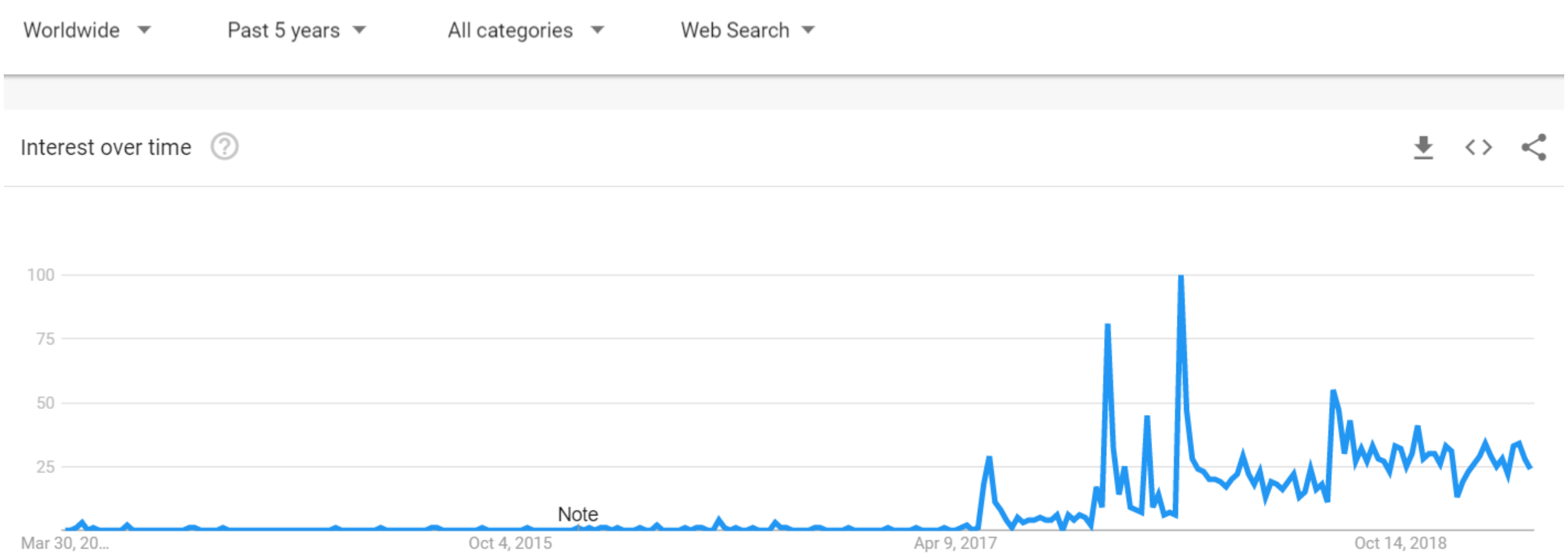
tpot - example

```
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)
tpot.fit(X_train, y_train)
print(tpot.score(X_test, y_test))
tpot.export('tpot_mnist_pipeline.py')
```

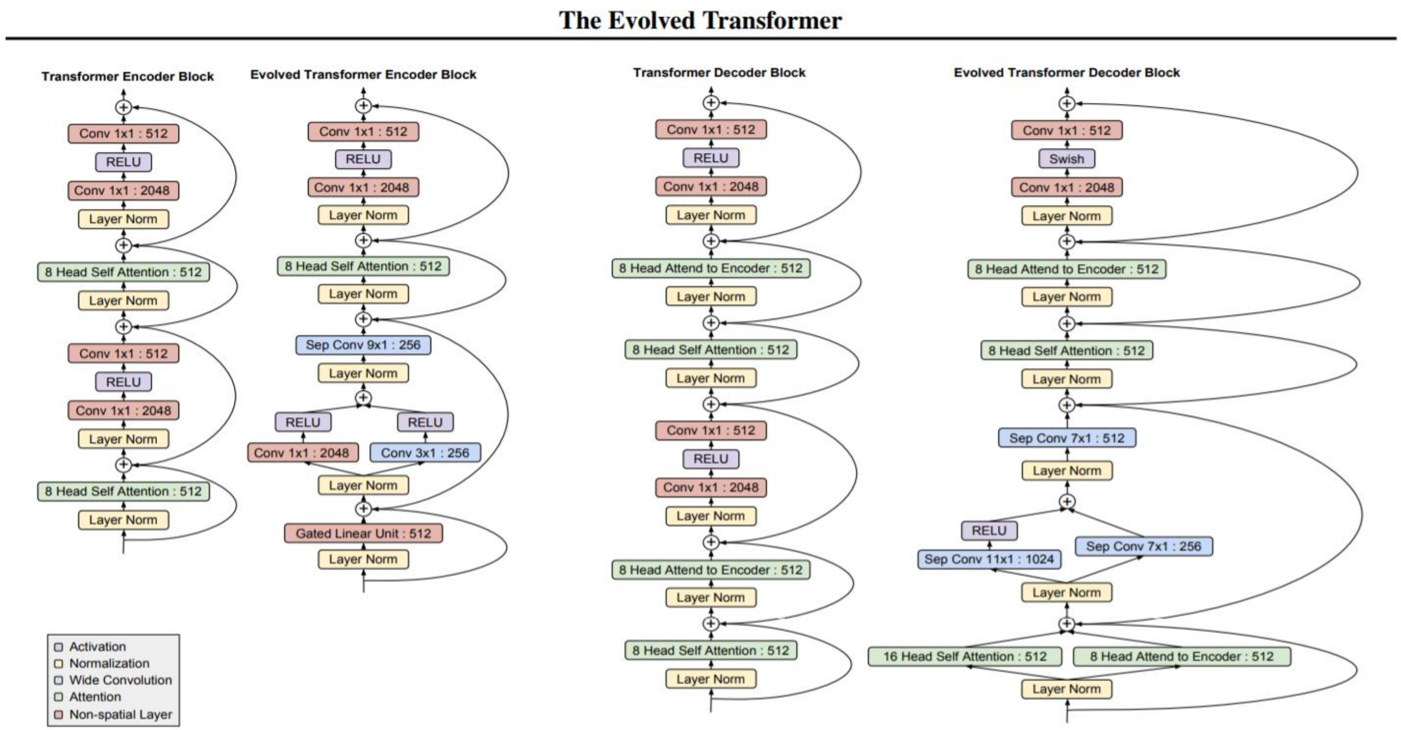
Progress of AutoML



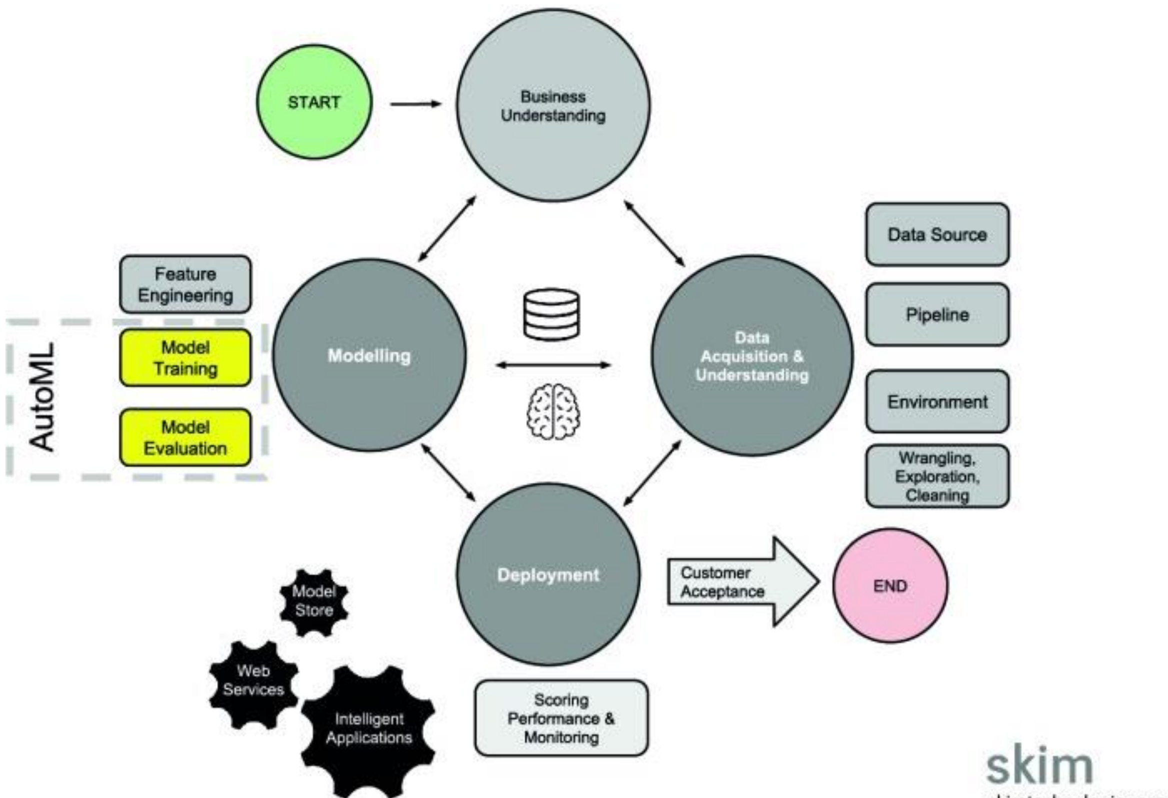
AutoML Vs Data Scientists

- Narrow AI engineers
- Domain expertise
- Exploit transfer learning

AutoML Vs Researchers



Future of AutoML



Freiburg AutoML group

www.ml4aad.org

Thanks for coming!

Feedback!

https://tinyurl.com/pydata-feedback

Slides

https://github.com/pydatamumbai

https://github.com/bhavsarpratik/automl

Telegram group

https://tinyurl.com/pydata-telegram

My blog

ml-dl.com

Connect

twitter.com/pratikbhavsar11

linkedin.com/in/bhavsarpratik

Genetic algorithms

- Genetic Algorithms properties:
  - **Selection**: You have a population of possible solutions to a given problem and a fitness function. At every iteration, you evaluate how to fit each solution with your fitness function.
  - **Crossover**: Then you select the fittest ones and perform crossover to create a new population.
  - **Mutation**: You take those children and mutate them with some random modification and repeat the process until you get the fittest or best solution.

START:

Step 1: ***Initialize***. Create a population of N elements, each with randomly generated DNA.

LOOP:

Step 2: ***Selection***. Evaluate the fitness of each element of the population and build a mating pool.

Step 3: ***Reproduction***. Repeat N times:

a) Pick parents with probability according to relative fitness.
b) Crossover—create a “child” by combining the DNA of these parents.
c) Mutation—mutate the child’s DNA based on a given probability.
d) Add the new child to a new population.

Step 4. Replace the old population with the new population and return to Step 2.

https://natureofcode.com/book/chapter-9-the-evolution-of-code/

HPO(Hyper-parameter Optimisation)

- Hyperopt, including the TPE algorithm
- Sequential Model-based Algorithm Configuration (SMAC)
- Spearmint
- BOHB: Bayesian Optimization combined with HyperBand
- RoBO – Robust Bayesian Optimization framework
- SMAC3 – a python re-implementation of the SMAC algorithm

Comparison – House Prices Challenge

https://hackernoon.com/a-brief-overview-of-automatic-machine-learning-solutions-automl-2826c7807a2a