AutoML: Hyperparameter Optimization

Preprocessing

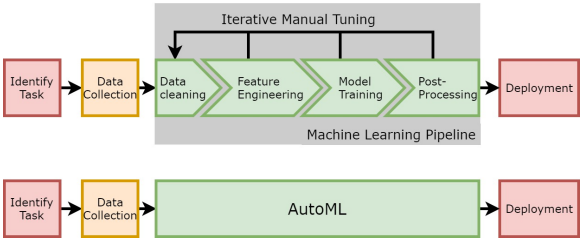
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer

Preprocessing

Ideal AutoML systems should also optimize:

- ✗ Data preprocessing
- ✗ Feature engineering
- ✗ Feature selection
- ✓ Model training

✓ = already covered, ✗ = not covered so far



Preprocessing capabilities differ heavily

Tool	Platform	Input data sources		Data pre-processing	Data types detected					Feature engineering					ML Tasks		Model selection and Hyperparameter optimization					Quick start / early stop			Model evaluation / Result analysis/ Visualization		
		Spreadsheet datasets	Image, text		Numerical	Categorical	Datetime	Time-series	Other (Hierarchical types) (7*)	Datetime, categorical processing	Imbalance, missing values	Feature selection, reduction	Advanced feature extraction (6*)		Supervised learning (9*)	Unsupervised learning (10*)	Ensemble	Genetic algorithm	Random search	Bayesian search	Neural architecture search	Quick finding of starting model	Allow maximum limit search time	Restrict time consuming combination of components		Model dashboard	Feature importance	Model explainability and interpretation, and reason code (11*)
TransmogrifAI	Apache Spark	Y	N	Y(7)	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	N	Y	N	Y	Y	N	N				Y	Y	
H2O-AutoML	H2O clusters	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	N		Y	N	Y	N	Y	N	N	N	Y	Y		Y	Y	Y
Darwin (+)	GCP	Y	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y		Y	Y	Y	Y	N	N	Y	Y	Y	N		Y	Y	Y
DataRobot (+)	Datarobot & AWS	Y	Y	Y	Y	Y	N	Y	N	Y	Y	Y	Y		Y	Y	Y	Y	Y	N	N	N	Y			Y	Y	Y
Google AutoML (+)	Google Cloud	N	Y	Y						N	Y	Y	Y		Y		Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y
Auto-sklearn		Y	N	N	N	N	N	N	N	Y(2*)	Y	Y	Y		Y	N	Y	N	Y	Y	N	Y	Y	Y		Y	Y	Y
MLjar (+)	MLJAR Cloud	Y(3*)	N	Y	Y	Y	N	N	N	Y	Y(4*)	N	N		Y(5*)	N	Y	N	Y	N	N	N	N	N		Y	Y	N
Auto_ml		Y	N	N	N	N	N	N	N	Y	Y	Y	Y		Y	N	Y	N	Y	Y	N	N	N	N		Y	Y	Y
TPOT		Y	N	N	N	N	N	N	N	N	Y	N	Y		Y	N	Y	Y	N	N	N	N	Y	N		Y	Y	N
Auto-keras		Y	Y	N	N	N	N	N	N	N	Y	Y	N		Y	N	N	N	Y	Y	Y	Y	Y	N		Y	N	Y
Ludwig		Y	Y	Y(7)	Y	Y	N	Y	Y	N	Y	Y	Y		Y	N	Y	N	Y	Y	Y	Y	N	N		Y	Y	N
Auto-Weka		Y	N	N	Y	Y	N	N	N	N	Y	Y	N		Y	N	Y	N	Y	Y	N	N	Y	Y		Y	N	N
Azure ML (+)	Azure	Y	Y	Y(8*)	Y	Y	Y	Y	N	Y	Y	Y	Y		Y	N	Y	N	Y	N			Y	Y		Y	Y	
Sagemaker (+)	AWS	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y		Y	N	Y	Y	Y	N			Y	N		Y	Y	Y
H2O-Driverless AI (+)	H2O clusters	Y(3*)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	N	N	N	Y	Y		Y	Y	Y

Fig. 2. Comparison table of functionality for AutoML tools. (+): commercialized tools; (∗): the function is not very stable, it fails for some datasets; (2∗): categorical input must be converted into integers; (3∗): datasets have to include headers; (4∗): missing values must be represented as NA; (5∗): multiclass classification not provided; (6∗): need some users' input for dataset description such as column types; (7∗): ability to detect primitive data types and rich data types such as: text (id, url, phone), numerical (integer, real); (8∗): advanced feature processing: bucketing of values, removing features with zero variance or features with drift over time; (9∗): supervised learning includes binary classification, multiclass classification, regression; (10∗): unsupervised learning includes clustering and anomaly detection; (11∗): model interpretation and explainability refers to techniques such as LIME, Shapley, Decision Tree Surrogate, Partial Dependence, Individual Conditional Expectation, Lift chart, feature fit, prediction distribution plot, accuracy over time, hot spot and reason codes; In a few empty cells, it is not clear that the functionality is provided from documentations of the tools, to the best of our knowledge.

Source [Truong et al. 2019].

Highlighted: Non-commercial AutoML frameworks

- Auto-detection of feature types: some
- Preprocess categoricals: some
- Imputation: all
- Class imbalance handling: all

Cleaning

Data cleaning is hard to fully automate, but a few heuristics exist:

- Remove ID columns, columns with mostly unique values.
- Outlier detection
- Detect time series or spatial data $\rightarrow$ at the very least, evaluation and resampling needs to be adapted, but feature extraction likely as well

It is highly unclear how much of this is required input by the user (last point is more or less task specification), and what should be automated by the system.

Categorical Features: Dummy Encoding

- Most simple encoding
- Works well with low cardinality categoricals

SalePrice	Central.Air	Bldg.Type
189900	Y	1Fam
195500	Y	1Fam
213500	Y	TwnhsE
191500	Y	TwnhsE
236500	Y	TwnhsE

SalePrice	Y	Bldg.Type.2fmCon	Bldg.Type.Duplex	Bldg.Type.Twnhs	Bldg.Type.TwnhsE
189900	1	0	0	0	0
195500	1	0	0	0	0
213500	1	0	0	0	1
191500	1	0	0	0	1
236500	1	0	0	0	1

Categorical Features: Target / Impact Encoding

- Well known from CART
- Handles high cardinality categoricals, too

$\mathcal{D}_{\text{train}}$

x	y
A	10
A	20
B	30

Encoding

x	$\tilde{x}$
A	-5
B	10

$\mathcal{D}_{\text{test}}$

x	$\tilde{x}$
A	-5
B	10
C	0

Goal: Encodes each categorical $\boldsymbol{x}$ as a single numeric $\tilde{\boldsymbol{x}}$

$$\text{Regression: } \operatorname{Impact}(x) = \mathbb{E}(\boldsymbol{y}|x) - \mathbb{E}(\boldsymbol{y})$$

$$\text{Classification: } \operatorname{Impact}(x) = \operatorname{logit}(P(y = \text{target}|x)) - \operatorname{logit}(P(y = \text{target}))$$

- Needs regularization (through CV) to prevent target leakage [Zumel et al. 2019]
- Advantage: Handles unknown categorical levels on test data

Categorical Features: Target / Impact Encoding

- Well known from CART
- Handles high cardinality categoricals, too

$\mathcal{D}_{\text{train}}$

x	y
A	10
A	20
B	30

Encoding

x	$\tilde{x}$
A	-5
B	10

$\mathcal{D}_{\text{test}}$

x	$\tilde{x}$
A	-5
B	10
C	0

Goal: Encodes each categorical $\boldsymbol{x}$ as a single numeric $\tilde{\boldsymbol{x}}$

$$\text{Regression: } \text{Impact}(x) = \mathbb{E}(\boldsymbol{y}|x) - \mathbb{E}(\boldsymbol{y})$$

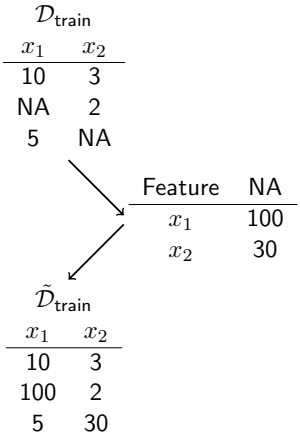
$$\text{Classification: } \text{Impact}(x) = \text{logit}(P(y = \text{target}|x)) - \text{logit}(P(y = \text{target}))$$

- Needs regularization (through CV) to prevent target leakage [Zumel et al. 2019]
- Advantage: Handles unknown categorical levels on test data

Alternatives:

- factorization machines
- clustering feature levels
- feature hashing

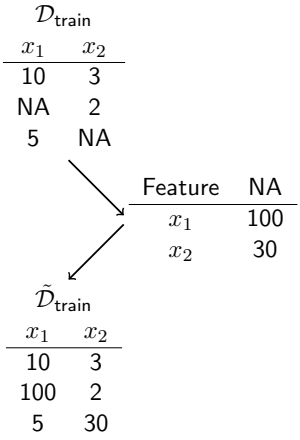
Common Preprocessing Steps: Missing Values



- Replace missings with imputed values, try not to disturb distribution too much
- Examples: mean, median, mode, sample from histogram, predict value based on other features

Out of range imputation

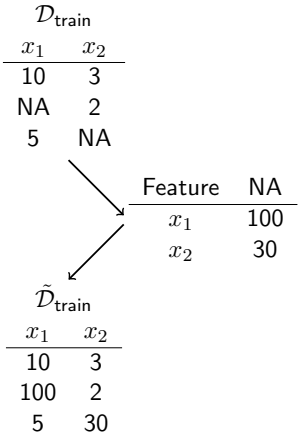
Common Preprocessing Steps: Missing Values



Out of range imputation

- Replace missings with imputed values, try not to disturb distribution too much
- Examples: mean, median, mode, sample from histogram, predict value based on other features
- Additional factor column indicating missingness can help if NA-state carries information for target
- *Out-Of-Range* works often well for tree-based techniques (RF and boosting!), saves extra missingness column

Common Preprocessing Steps: Missing Values

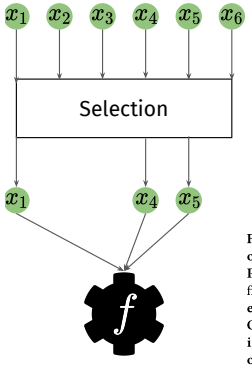


Out of range imputation

- Replace missings with imputed values, try not to disturb distribution too much
- Examples: mean, median, mode, sample from histogram, predict value based on other features
- Additional factor column indicating missingness can help if NA-state carries information for target
- *Out-Of-Range* works often well for tree-based techniques (RF and boosting!), saves extra missingness column
- For inference, simple imputation is often shunned, and multiple, model-based imputation recommended; for prediction naive imputation often works remarkably well

Feature Selection

- Filter; for extremely high dim.
- Stepwise / wrapper methods; seldom increases performance when well-regularized learners are used, but quite expensive
- Embedded: CART, lasso, ...
- Selection interesting when predictive performance vs. sparseness should be optimized $\rightarrow$ multi-criteria optimization
- Combined feature selection and HPO: [Binder et al. 2020]



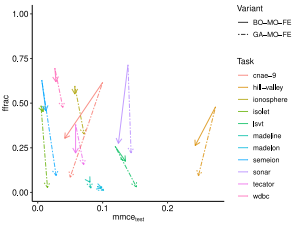


Figure 3: Comparison of multi-objective and single-objective methods applied to the SVM learning algorithm: Performance $\text{mmce}_{\text{test}}$ and the fraction of included features ffrac found after 2000 evaluations by baseline BO-SO (tail end of arrows), BO-MO-FE (head of solid arrows) and GA-MO-FE (head of dashed arrows). Each dataset (Table 1) is shown, values are averaged over 10 outer CV runs. Choice of individuals for MO methods described in Section 6.2.

Source: [Binder et al. 2020].

Common Feature Construction Methods

Reduction:

- PCA, ICA, autoencoder

Feature extraction:

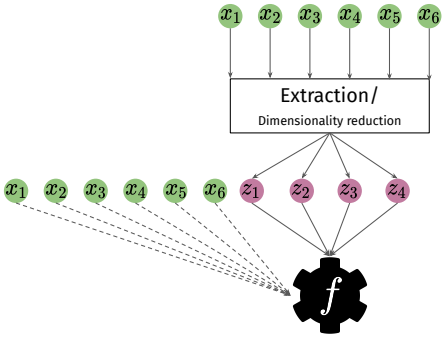
- Polynomial features: $x_j \longrightarrow x_j, x_j^2, x_j^3, ...$
- Interactions: $x_j, x_k \longrightarrow x_j, x_k, x_j \cdot x_k$

Feature generation:

- Transform to "circular" features (month, day)
  e.g. $\tilde{x}_1 = sin(2\pi \cdot x/24)$ and $\tilde{x}_2 = cos(2\pi \cdot x/24)$

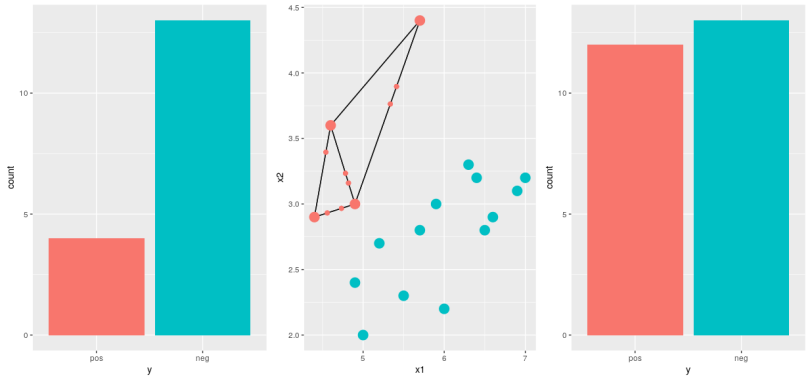
Combine with external data:

- names $\longrightarrow$ gender, ethnicity, age
- home address $\longrightarrow$ household income
- location + date $\longrightarrow$ weather



Imbalanced Classes

- Oversampling of minority class
- Seldom: undersampling of majority class
- Intelligent oversampling strategies which create synthetic observations (SMOTE)



Lecture Overview

1. Practical Problems
2. Combined Preprocessing and Model Building: Pipelining

Practical Problems: Stability

AutoML system should:

- never fail to return a result
- terminate within a given time
- save intermediate results and allow to continue

Failure points:

- optimizer can crash (e.g. GP in BO)
- training can crash (e.g. segfault)
- training can run "forever"

Ways out

- Encapsulate train/predict in separate process from HPO
- Ressource limit time and memory of that process by OS
- If learner crashes, run robust fallback (constant predictor)
- Use "robust" HPO, run random config as last resort if proposal fails (exploration)

Practical Problems: Parallelization

Parallelization should allow:

- multiple cores
- multiple nodes

Possible parallelization levels:

- training of learner (threading / GPU)
- resampling
- eval configurations (batch proposal of HPO)

Possible problems:

- Sequential nature of HPO algorithms (e.g. BO)
- Heterogeneous runtimes cause idling $\longrightarrow$ asynchronous HPO attractive, but more complex
- Main memory or CPU-cache becomes bottleneck
- Communication between workers

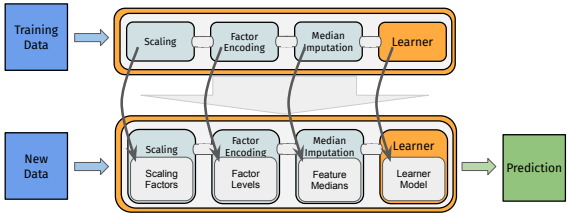
Lecture Overview

1. Practical Problems

2. Combined Preprocessing and Model Building: Pipelining

Linear Pipelining

- Applying preprocessing to the whole dataset leads to data leakage
- Preprocessing should have train and predict steps, too
- Can add it to learner, and embed it in CV
- Surprise: Preproc has hyperparameters
- Optimize pipeline jointly: $\mathbf{\Lambda} = \mathbf{\Lambda}_{\text{preproc}} \times \mathbf{\Lambda}_{\mathcal{I}}$
- Still HPO, not much different than for single learner
- $\mathbf{\Lambda}$ "simply" of higher dimension



Nonlinear Pipelines

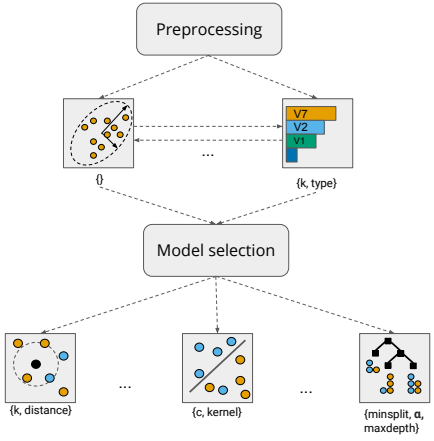
Ideal to let HPO choose automatically:

- preprocessing
- feature extraction
- learner

$\rightarrow$ $\mathbf{\Lambda}$ becomes hierarchical search space!

Suitable optimizers:

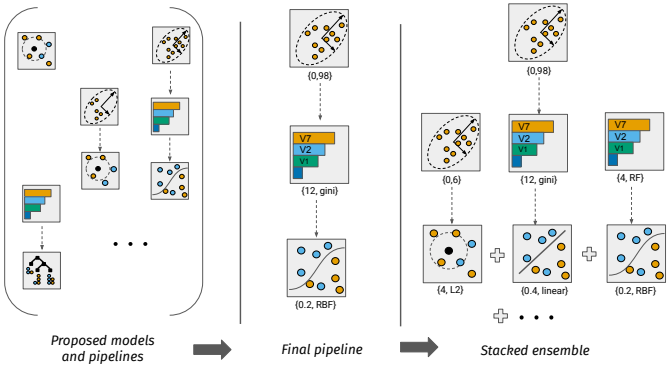
- TPE
- Random search, hyperband with sampler that follows the hierarchy
- BO with RF (imputation or hierarchical trees)
- Evolutionary approaches (similar to NAS)



Obtaining Final Model

Options:

- Choose the optimal path as linear pipeline.
- Build ensemble of best configurations
  (e.g. [Feurer et al. 2015], [LeDell and Poirier. 2020]).



Current Benchmark on Tabular Data

Framework: Binary tasks:	auto-sklearn	Auto-WEKA	H2O AutoML	RandomForest	TPOT
adult	1.045	1.000	1.049	1.000	1.048
airlines	1.403	1.016	1.435	0.997	1.343
albert	1.009		1.115	1.001	0.981
amazon_employee...	0.972*	0.886	1.048	1.003	1.012
apsfailure	1.000	0.985	1.001	1.000	1.001
australian	1.010	1.015	0.909	1.010	1.011
bank-marketing	1.012	0.950	1.015	1.000	1.008
blood-transfusion	1.495	1.379	1.532	0.985	1.149
christine	1.072	0.998	1.048	0.988	1.029
credit-g	0.970*	0.829	0.991	1.004	0.924
guillermo	1.004	0.934	1.024	0.999	0.878
higgs	1.018*	0.845	1.041	0.999	1.005
jasmine	0.987	0.939	1.001	0.998	1.004
kc1	0.999*	0.934	0.992	0.987	1.013
kddcup09_appetency	1.181*	1.043	1.176	1.016	1.134
kr-vs-kp	1.000*	0.959	1.000	0.999	0.999
miniboone	1.008	0.957	1.010	0.999	1.001
nomao	1.002	0.973	1.002	1.000	1.001
numerai28.6	1.679	1.544	1.730	1.042	1.428
phoneme	0.993*	0.998	1.005	1.000	1.015
riccardo	1.000	0.996	1.000	0.999	0.992
sylvine	1.013	0.985	1.011	0.999	1.023
Multi-class tasks:					
car	1.030	0.906	1.060	0.878	1.060
cnae-9	1.069	0.541	1.076	0.999	1.057
connect-4	1.184	-1.565	1.409	0.954	1.276
covertype	0.976	-0.361	0.856	0.944	0.933
dilbert	1.182	0.459	1.205	0.979	1.111
dionis	0.580	0.590		1.002	
fabert	1.026	-5.235	1.049	1.004	1.005
fashion-mnist	0.995	0.717	1.052	0.993	0.841
helena	0.660	-18.420	1.905	0.970	1.676
jannis	1.083	-1.989	1.065	0.973	0.987
jungle_chess...	1.299	-3.309	1.235	0.933	1.459
mfeat-factors	1.059*	0.789	1.053	0.992	1.018
robert	-0.001		1.545	1.000	0.640
segment	1.004	0.808	1.012	0.992	1.008
shuttle	1.000	0.979	1.000	1.000	1.000
vehicle	1.102	-4.630	1.166	0.986	1.099
volkert	1.002	-5.585	1.111	0.954	0.945

Table 2: Performance of AutoML frameworks, scaled between a constant class prior predictor (=0) and a tuned random forest (= 1). Missing values mean that no results were returned in time. *: the task was also included in meta-learning models.

Table taken from [Gijsbers et al. 2019].

- On some datasets AutoML yields big performance improvements
- On many datasets RF is equally good
- Need more and diverse benchmarks