AutoML: Hyperparameter Optimization

Preprocessing

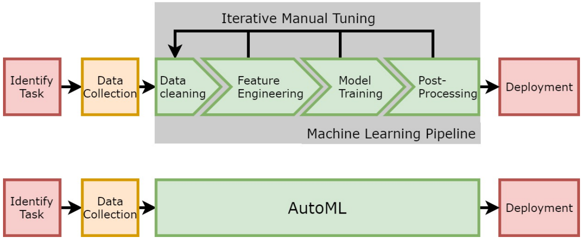
Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer

Preprocessing

An AutoML systems should also optimize:

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

✓ = already covered, ✗ = not covered so far



Simple Preprocessing and Cleaning

- Data cleaning is hard to fully automate since errors in the data can be semantic.
- A few simple things should always be done:
  - Remove ID columns, columns with only unique values
  - Remove duplicated columns
  - Remove constant columns
- Already harder, but still possible:
  - Detect outliers
  - Detect correct column types (numeric, discrete, datetime, ...)
  - Detect missing value encoding
- Even harder:
  - Detect spelling and formatting errors
  - Detect inconsistencies (e.g. zip code does not match city name)
  - Detect time series or spatial data

It is 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.

Preprocessing: Categorical Features

Categorical features have a fixed number of distict (unordered) possible values called levels.

- For most learners categorical features need to be encoded in numeric features.
- Distinguish between binary, low cardinality and high cardinality categorical features. Low or high cardinality refers to the number of levels.
  - Binary: Encode as $1 - 0$.
  - Low-cardinality: One-hot encoding.
  - High-cardinality: Regularized target/impact encoding, clustering, hashing.
- Tree-based algorithms (in some software implementations) can natively handle even high-cardinality categorical features.
- Optimal encoding can vary between each feature, algorithm and hyperparameter configuration. Introduce and tune threshold hyperparameter that decides when to use high-cardinality encoding, e.g. $n_{lvls} \geq \tau_{\text{high card.}}$.
- The encoder should also be able to handle new feature levels occuring at test time without crashing.

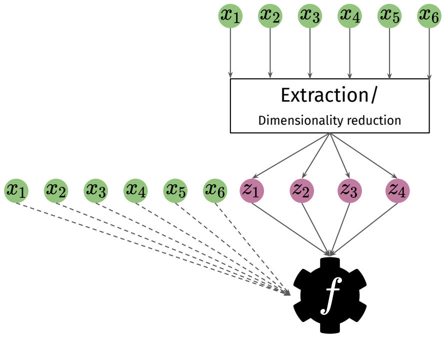
Preprocessing: Missing Values

Imputation is the process of replacing missing values with artificial substituted values.

- Simple imputation techniques replace missings with the mean, median, mode or a sample from empirical distribution of the feature.
- To keep track of the imputation, binary indicator features are added.
- For categorical features, missing values can easily be replaced by a new seperate level.
- Tree-based algorithms (in some software implementations) can natively handle missing values.
- Model-based imputation trains a machine learning model $f(x_{-i}) = x_i$ to predict missing values of x_i using the remaining features $x_{-i} = x_1, ..., x_{i-1}, x_{i+1}, ..., x_p$.
  - The imputation model should be able to handle missings natively.
  - The choice of learner and its hyperparameters add additional complexity to the imputation.
  - Random Forests are a reasonable choice.

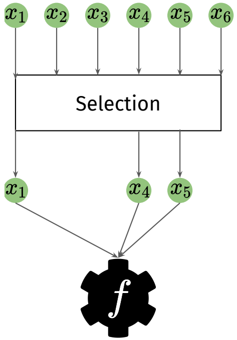
Feature Engineering

- $1:1$ Transformations
  - $\log$, $\sqrt{\cdot}$, ...
  - Standardization, Box-Cox, 0-1-scaling, ...
  - Binning of numeric features
- $1:many$ Transformations
  - Polynomial expansion: $x_j \longrightarrow x_j, x_j^2, x_j^3, \ldots$
  - Basis expansions with B/P-splines
  - Transform to "circular" features (month, day) e.g. $\tilde{x}_1 = sin(2\pi \cdot x_1/24)$
  - Extract Weekday/Day/... from datetime features
- $many:many$ Transformations
  - (kernel) Principal component analysis
  - Truncated singular value decomposition
  - Idependent component analysis
  - Autoencoder



Feature Selection

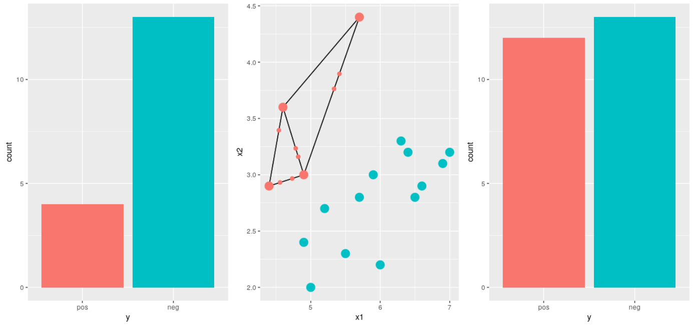
- Feature filter rank features and select most important fraction.
- Stepwise / wrapper methods greedily select one feature after another.
- Some learner have embedded feature selection CART, lasso, . . .
- p-dimensional bitvector as hyperparameters.
- Combined feature selection and HPO: [Binder et al. 2020]



Imbalanced Classes

Imbalanced classifcation occurs if one (usually more important) class appears much less frequent than other classes.

- Tune class weights as hyperparameters.
- Oversampling of minority class or undersmpling of majority class.
- Generate synthetic samples that (should) belong to the minority class, e.g. SMOTE.
- Be careful with resampling and ensure stratification.
- If imbalance is severe change framing, e.g. anomaly detection.


count
pos
neg
y
x2
x1
count
pos
neg
y