University of Ljubljana, Faculty of Computer and Information Science

Automated Machine Learning (AutoML)

Prof Dr Marko Robnik-Šikonja

Intelligent Systems, Edition 2025

Why AutoML?

- Growing demand for ML expertise
- Manual ML pipelines are time-consuming
- Evolution: Manual ML → Automated ML
- End-to-end automation of ML pipeline
- AutoML democratizes AI
- Use cases in competitions, industry, etc.
- Not a magic

Traditional ML Pipeline

- Steps: data→preprocessing→model→evaluation→deployment
- AutoML automates most parts
- Instead of manually choosing:
  - features
  - models
  - hyperparameters
  - training strategies

AutoML systems automatically search, optimize, and adapt.

Automated Preprocessing

- Missing data handling
- Scaling
- Feature construction

Feature engineering automation

- Automatically generate, transform, or select features
- Automate:
  - normalization & scaling
  - categorical encoding
  - feature crosses
  - polynomial features
  - feature selection
- Approaches
  - rule-based pipelines
  - evolutionary algorithms
  - attention-based feature learning
  - embedded methods (e.g., L1 regularization)
- Example: create log(income), income/age, select top-k features

Model selection

- Automatically choose *which algorithm* to use.
- Search space includes: linear models, tree-based models, kernel methods, neural networks, etc.
- Techniques
  - Bayesian optimization over models
  - ensemble selection
  - meta-learning (use prior tasks to guide choice)
- Example: Should I use Random Forest, XGBoost, or a Neural Network?

Meta-Learning (learning to learn)

- Use experience from previous datasets to speed up AutoML.
- Store dataset “meta-features” (size, skewness, entropy, etc.)
- Learn which models worked best before
- warm-start optimization
- Example: if the dataset looks like past credit-risk data → try XGBoost first
- This reduces search time drastically.

Neural architecture search (NAS)

- Automatically design neural network architectures.
- Search space:
  - number of layers
  - width
  - skip connections
  - attention vs convolution
- Search strategies
  - reinforcement learning
  - evolutionary algorithms
  - gradient-based NAS (e.g., DARTS)
- Example: find the best CNN or Transformer architecture.

Hyperparameter optimization

- Automatically finds good hyperparameters for a fixed model
- This is the most common and mature AutoML technique.
- Typical parameters: learning rate, number of layers, tree depth, regularization strength, etc
- Methods
  - Grid search (simple, inefficient)
  - Random search (surprisingly strong baseline)
  - Bayesian optimization (Gaussian Processes, TPE)
- Example: Find best (learning_rate, max_depth) for XGBoost

Random search

- Random search optimizes hyperparameters by sampling configurations randomly,
- This is more efficient than grid search in high-dimensional spaces because it focuses effort on important parameters naturally.
- Assuming that not all parameters are equally important, grid search only samples a few values of important parameters, while random search samples many more distinct values of important parameters with the same computational budget
- Grid search:
  - assume thet you use a grid with n values per parameter
  - Total $\textit{trials} = n^d$
  - Each parameter gets exactly n values, regardless of importance.
- Random search
  - Let k= number of important parameters, N= total trials
  - Random search explores $\mathcal{O}(N)$values per important parameter
- Grid search explores $\mathcal{O}\left(N^{1/k}\right)$values per important parameter
- For large d, this gap is enormous.

Pipeline Optimization (End-to-End AutoML)

- Optimize entire ML pipelines, not just components
- A pipeline might be: StandardScaler → PCA → XGBoost
- AutoML searches over:
  - preprocessing steps
  - feature selection
  - model choice
  - hyperparameters
- Key methods
  - Bayesian optimization
  - genetic programming
  - hierarchical search spaces
- Example systems
  - Auto-sklearn
  - TP ’s AutoML
  - H2O AutoML

Bayesian optimization

- Bayesian Optimization is a strategy for optimizing expensive, unknown, and non-convex functions using as few evaluations as possible.
- It is typically used when evaluating the objective is slow or costly (e.g., training a model), gradients are unavailable, the function may be noisy, and/or the number of evaluations is limited.
- In ML, the function is often:
  $$f(\theta) = \textit{validation performance of model with hyperparameters } \theta$$
- Bayesian optimization builds a surrogate probabilistic model of the objective function and uses it to decide where to evaluate next, balancing exploration and exploitation

Strengths and limitations of AutoML

- + Saves time and expertise
- + Strong baselines quickly
- + Reduces human bias
- + Reproducible workflows

- − Computationally expensive
- − Limited interpretability
- − Hard to encode domain knowledge
- − May overfit if not controlled
- − High compute cost