SCML 2026
International Conference on
Symbolic Computation and Machine
Learning

RISC, Johannes Kepler University Linz

July 6–8, 2026
Hagenberg, Austria

Learning to Rank Symbolic Expressions for Model Selection

Can a learned meta-model beat classical criteria for model selection?

Ali Soltani[1], *Gabriel Kronberger*[2], *Fabricio Olivetti de França*[3], *Alessandro Lucantonio*[1]
[1]*Aarhus University*, [2]*University of Applied Sciences Upper Austria*, [3]*Federal University of ABC*

Presenter: Ali Soltani | *asoltani@mpe.au.dk*

Why model selection matters in symbolic regression

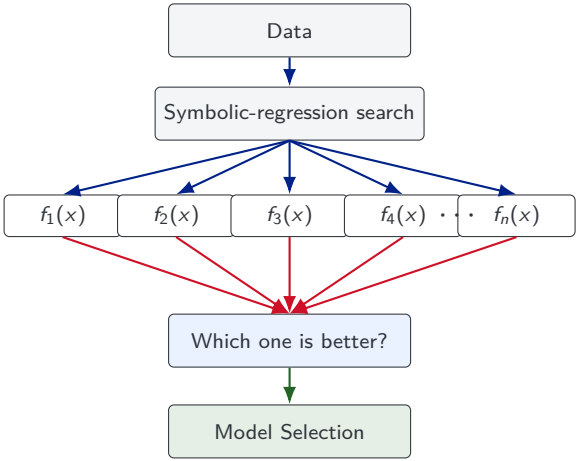
From machine learning to symbolic regression

Machine learning usually searches for a model that predicts well from data.

Symbolic regression goes one step further: it searches for an explicit mathematical expression

$$\hat{y} = f(x_1, \ldots, x_d)$$

which can be both accurate and interpretable.



Symbolic regression does not end with one equation.

Candidate A
compact
higher training error

Candidate B
larger structure
lower training error

Candidate C
excellent fit
likely overfitted

The **search algorithm** proposes **candidates**. The **selection criterion** decides which one to **choose**.

Symbolic regression

Goal and output

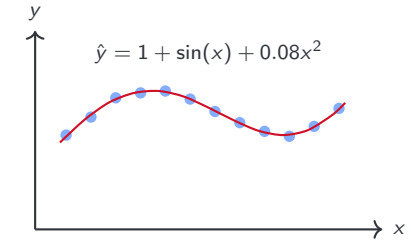
Symbolic regression searches for an interpretable expression

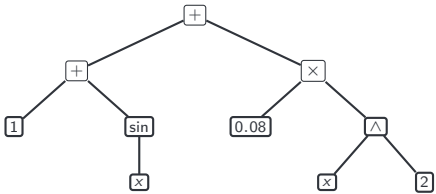
$$\hat{y} = f(x_1, \ldots, x_d; \boldsymbol{\theta})$$

that explains a target y from data.
The expression may contain nonlinear operators, constants, and can be represented as an expression tree.

Why it matters

- Capability of producing compact, interpretable formulas
- Useful in science and engineering





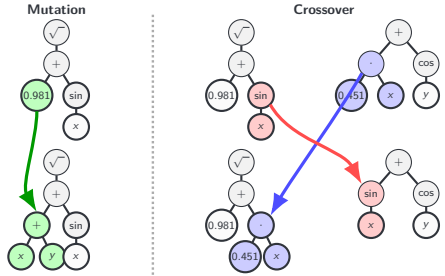
Genetic programming for symbolic regression

Search mechanism

GP is an iterative population-based search:

- start from a population of expression trees
- create new trees via crossover, mutation, and reproduction
- evaluate candidates using a fitness function
- keep high-quality trees and discard weaker ones
- repeat until a stopping criterion is reached

Genetic operators on expression trees



What GP optimizes during search

Usually GP optimizes training error, sometimes with model size or structural complexity:

$$\text{fitness} \approx \text{MSE}_{\text{train}} + \text{complexity penalty}$$

Pareto fronts: accuracy vs. complexity

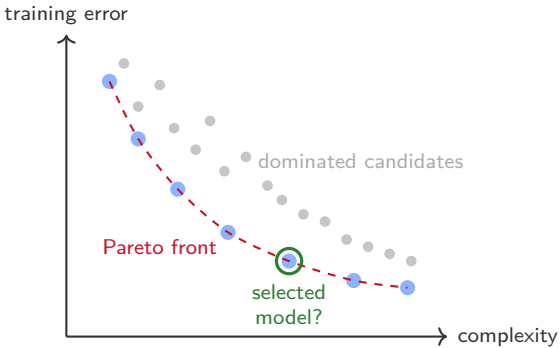
Multi-objective GP

Each run keeps non-dominated models:

- low training error
- short program length / low complexity

Our data object

Each **Pareto front** is a set of competing expressions from one GP run on one dataset split.



Model selection = pick the right point on the front

Model-selection criteria

Criterion	Selection principle
MSE_{train}	training fit only
Akaike Information Criterion (AIC)	likelihood + parameter penalty
Bayesian Information Criterion (BIC)	likelihood + sample-size based parameter penalty
Description length (DL)	likelihood + structure + parameter complexity
Fractional Bayes Factor (FBF)	likelihood + structure + parameter complexity

Motivation

DL and FBF beat the two other methods in selection from GP Pareto fronts. Our question is whether a **learned ranker** can combine these signals better than any single criterion?

Query id and relevance labels

What is a qid?

A `qid` defines one independent ranking group.

In our case:

$$\texttt{qid} \equiv \text{one GP Pareto front}$$

Example

Dataset / run	Candidate	qid
Salustowicz, run 1	$f_1(x)$	17
Salustowicz, run 1	$f_2(x)$	17
Salustowicz, run 1	$f_3(x)$	17
Nikuradse, run 1	$g_1(x)$	42
Nikuradse, run 1	$g_2(x)$	42

What is relevance?

Relevance is the target ranking label used by the learning-to-rank model.

Inside each `qid`, we sort candidates by held-out test error:

$$\text{lower } \texttt{MSE\_test} \quad \Rightarrow \quad \text{higher relevance}$$

Example within one qid

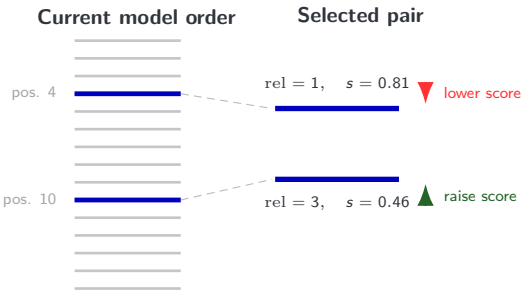
Candidate	MSE_test	Rank r_i	Relevance
$f_2(x)$	0.012	1	3
$f_1(x)$	0.030	2	2
$f_3(x)$	0.080	3	1

Pairwise ranking objective

Within each `qid`, the model compares pairs of candidates:

- take two expressions from the same Pareto front
- check which one should be ranked higher
- compare their current model scores
- penalize the model if the better expression is scored lower
- convert each pairwise mistake into an update direction

Pairwise comparison inside one `qid`



The lower item has higher relevance, but the model currently gives it a lower score. The pairwise objective therefore pushes this item up and the other one down.

Dataset: GP Pareto fronts

Source

Pareto-front logs from multi-objective GPSR runs using NLL and expression length as objectives Kronberger et al., 2026.

Dataset groups

- **Synthetic**: controlled ground truth, noise/sample-size studies
- **Engineering**: Nikuradse, friction, and chemical-process datasets
- **SRBench selection**: real regression benchmarks

Scale after cleaning

- **286,736** candidate expressions
- **3,400** query groups: 34 problem instances $\times$ 100 runs
- each `qid` is one GP Pareto front
- each row stores expression, criteria, scores, and metadata

Evaluation target

Final model quality is measured by held-out `MSE_test`.

Cross validation

Repeated family-level splits

We run the full algorithm **5 times**. In each run, we randomly split by **problem family**:

- 20% families for test
- 20% families for validation
- remaining families for training

Why split by family?

Rows are not split independently.

All `qid` groups belonging to the same problem family stay in the same split.

Evaluation

Within each run, scores are averaged across all `qids` in the split.

Final reported results are averaged across the **5 runs**.

Evaluation metric: Top-k hit rate

Idea

For each `qid`, we ask:

Did the method place the truly best models near the top?

Formula

For group g:

$$\mathrm{HitRate@}k = \frac{|\,T_g^{(k)} \cap P_g^{(k)}|}{k}$$

where $T_g^{(k)}$ is the true top-k set and $P_g^{(k)}$ is the predicted top-k set.

Example: Top-3

Candidate	True rank	Pred. rank
f_1	1	2
f_2	2	5
f_3	3	1
f_4	4	3
f_5	5	4

$T^{(3)} = \{f_1, f_2, f_3\}$ $\quad$ $P^{(3)} = \{f_3, f_1, f_4\}$

$$T^{(3)} \cap P^{(3)} = \{f_1, f_3\}$$

$$\mathrm{HitRate@}3 = \frac{2}{3}$$

Result: Top-k hit rate on the test split

Method	top-1	top-3	top-5	top-10
Ranker	**0.0992**	**0.2012**	**0.2817**	**0.4043**
MSE_{train}	0.0091	0.0221	0.0308	0.0499
AIC	0.0080	0.0204	0.0325	0.0517
BIC	0.0448	0.1007	0.1432	0.2386
DL	0.0889	0.1730	0.2460	0.3692
FBF	0.0879	0.1688	0.2368	0.3647

Takeaway

On unseen test families, **Ranker** achieves the best top-k recovery across all cutoffs, followed by DL and FBF.

Summary

Problem

Choose the best symbolic model from each GP Pareto front.

Goal

Build a data-driven ranker for symbolic expressions.

Data

Diverse symbolic problems including different levels of noise and expressions.

Result

Ranker performs slightly better than DL/FBF and clearly better than AIC/BIC.

Takeaway

Learning-to-rank is a promising meta-model strategy for selecting symbolic expressions from GP Pareto fronts.

Future work

Next steps

- richer feature sets: residuals, structure, and data summaries
- stronger cross-validation over dataset families and folds
- hybrid selectors: learned ranker + DL/FBF post-processing
- per-dataset analysis of where learning beats classical criteria

Thank you!

LinkedIn

Supplementary slides

For Q&A and detailed discussion

Backup: Criterion formulas

$$\mathrm{NLL} = -\log \mathcal{L}(\hat{\theta}, \hat{\sigma}^2_{\mathrm{err}})$$

Information criteria

$$\mathrm{AIC} = 2\,\mathrm{NLL} + 2p$$

$$\mathrm{BIC} = 2\,\mathrm{NLL} + p \log m$$

Description length

$$\mathrm{DL} = \mathrm{NLL} + \mathrm{funccomp} + \frac{1}{2} \sum_{i=1}^{p} \max\left(0,\ \log S_{ii} - \log 3 + \log |\hat{\theta}_i^{\mathrm{rot}}|\right).$$

Function complexity

$$\mathrm{funccomp} = k \log s + \sum_{c_i} \log c_i$$

Fractional Bayes factor

$$\mathrm{FBF} = (1-b)\,\mathrm{NLL} + \mathrm{funccomp} + \frac{p}{2}\left[\log\left(2\pi e^{1-\log 3}\right) - \log b\right], \qquad b = m^{-1/2}.$$

Symbols

m	number of training observations	p	number of fitted parameters
k	expression length	s	number of distinct symbols
c_i	constants encoded in function complexity	S_{ii}	FIM singular values
$\hat{\theta}^{\mathrm{rot}}$	rotated fitted parameters	$\mathcal{L}$	Gaussian likelihood at MLEs

Backup: pairwise and NDCG formulas

Pairwise surrogate

For candidate i ranked above j:

$$P_{ij} = \frac{1}{1 + \exp[-\sigma(s_i - s_j)]}$$

$$C_{ij}^{\text{pair}} = -\log P_{ij} = \log\Big(1 + \exp[-\sigma(s_i - s_j)]\Big)$$

$$\lambda_{ij}^{\text{pair}} = -\frac{\sigma}{1 + \exp[\sigma(s_i - s_j)]}$$

NDCG weighting

$$\text{DCG@}T = \sum_{r=1}^{T} \frac{2^{\ell_r} - 1}{\log(1 + r)}$$

$$\text{NDCG@}T = \frac{\text{DCG@}T}{\text{IDCG@}T}$$

$$\lambda_{ij}^{\text{NDCG}} = \lambda_{ij}^{\text{pair}} |\Delta\text{NDCG}_{ij}|$$

From lambdas to boosted trees

Pairwise forces are accumulated per candidate:

$$\lambda_i = \sum_{j:\{i,j\}\in I} \lambda_{ij} - \sum_{j:\{j,i\}\in I} \lambda_{ji}$$

Tree targets, leaf values, and update:

$$y_i = \lambda_i, \qquad w_i = \frac{\partial y_i}{\partial F_{m-1}(\mathbf{x}_i)}$$

$$\gamma_{\ell m} = \frac{\sum_{\mathbf{x}_i \in R_{\ell m}} y_i}{\sum_{\mathbf{x}_i \in R_{\ell m}} w_i}$$

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta \sum_{\ell} \gamma_{\ell m} \mathbb{I}(\mathbf{x} \in R_{\ell m})$$

Symbols

i, j	candidates in same qid	λ_{ij}	pairwise force	ΔNDCG_{ij}	swap effect
s_i, s_j	current scores	λ_i	total update signal	w_i	Hessian weight
σ	sigmoid steepness	I	ordered training pairs	$R_{\ell m}$	leaf ℓ of tree m
P_{ij}	probability $i > j$	ℓ_r	relevance at rank r	$\gamma_{\ell m}$	leaf value
C_{ij}	pairwise logistic loss	T	ranking cutoff	η	learning rate
y_i	tree pseudo-target	IDCG	ideal DCG	F_m	boosted model

Backup: datasets used in Kronberger et al.

Synthetic and engineering datasets

Dataset	Obs. (n_{train}, n_{test})	Vars.
Friedman-1	(50–1000, 1000)	10
Friedman-2	(100, 1000)	10
Kotanchek	(100, 2025)	3
RatPol2d	(50, 1156)	2
RatPol3d	(300, 2700)	3
Ripple	(300, 1000)	2
Salustowicz	(100, 220)	1
Salustowicz2d	(606, 2553)	2
Nikuradse 1d	(230, 132)	1
Nikuradse 2d	(201, 163)	2
Friction dyn.	(1009, 1007)	17
Friction stat.	(1009, 1007)	16
Chemical Tower	(3136, 1865)	25
Chemical Comp.	(711, 355)	57

SRBench selection

Dataset	Obs. (n_{train}, n_{test})	Vars.
192 Vineyard	(31, 21)	2
210 Cloud	(64, 44)	5
522 PM10	(300, 200)	7
557 Analcat Apnea1	(285, 190)	3
579 Fri. c0 250 5	(150, 100)	5
606 Fri. c2 1000 10	(600, 400)	10
650 Fri. c0 500 50	(300, 200)	50
678 Vis. env	(66, 45)	3
1028 SWD	(600, 400)	10
1089 USCrime	(28, 19)	13
1193 BNG lowbwt	(1000, 10368)	9
1199 BNG echo months	(1000, 5832)	9

Backup: Top-k hit rate on train and validation

Train split

Method	top-1	top-3	top-5	top-10
Ranker	0.1244	0.2684	0.3647	0.5134
MSE_{train}	0.0041	0.0087	0.0124	0.0255
AIC	0.0039	0.0088	0.0141	0.0289
BIC	0.0376	0.0864	0.1269	0.2148
FBF	0.1346	0.2684	0.3571	0.4888
DL	0.1277	0.2641	0.3502	0.4652

Validation split

Method	top-1	top-3	top-5	top-10
Ranker	0.0805	0.1862	0.2628	0.3694
MSE_{train}	0.0049	0.0076	0.0124	0.0342
AIC	0.0036	0.0072	0.0144	0.0396
BIC	0.0400	0.0866	0.1336	0.2297
FBF	0.1593	0.2896	0.3812	0.5004
DL	0.1484	0.2875	0.3713	0.4819