TABEVO: LLM-GUIDED EVOLUTION OF DATASET-CONDITIONAL PROGRAMS FOR TABULAR IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A tabular in-context learner is a function $f(D, X^*)$ from a labelled table and query rows to predictions, which tabular foundation models implement with a transformer pretrained on millions of synthetic tables. We show that the same function can be an explicit program found by search guided by a large language model (LLM). TabEvo represents a learner as a meta-DAG, a directed graph whose edges carry code that fits a transformation or a predictor on the rows of the table at hand, whose nodes merge the resulting blocks, and whose head combines prediction blocks by cross-fitted non-negative stacking, so that no component reads a prediction made with its own label. An LLM agent loop grows a program one accepted edit at a time, and an evolutionary search then edits it through parent-specific structured diffs; every child is fitted on 15 development tables before it enters a MAP-Elites archive. The resulting program has four exact Gaussian-process views of the input space, two residual learners, repeat-detecting gates and a gradient-boosted lane. Frozen and evaluated under a pre-registered protocol on 95 tables against 45 baseline configurations, and with no pretraining, it is within one point, by our equivalence rule, of seven of the nine foundation models on 22 of the 26 fresh TabArena tables, 0.7 to 0.9 points behind TabFM and LimiX, and level with bagged CatBoost, RealMLP and a Super Learner of trained models. On 23 of 24 fresh regression tables it is ahead of every trained model, ahead of the Super Learner by a margin that closes once a Gaussian process joins the stack, and about one point behind the newest foundation models. Because it is a program, it can be tested and edited. With two thousand mostly irrelevant features it keeps an R^2 of 0.86 where the foundation models fall to 0.33–0.39, and an automated edit-and-test loop halved its graph and made it 6.5$\times$ faster, at a 0.3-point cost in accuracy and no detected change in ROC-AUC.

1 INTRODUCTION

In-context learning changed what a tabular model is. A classical method is a recipe run anew on every dataset, whereas a tabular foundation model (FM) is one fixed function $f(D, X^*) \mapsto \widehat{Y}^*$ from the labelled table D and the query rows X^* to predictions. TabPFN and TabICL obtain f by pretraining a transformer on millions of synthetic tables (Hollmann et al., 2025; Qu et al., 2025; 2026) and now lead the benchmarks (Erickson et al., 2025). The price is that f is a tensor of weights. It cannot be read, and its behaviour outside the pretraining distribution is unknown; changing it means fine-tuning or pretraining its weights.

This paper implements the same function as a *program*, a directed acyclic graph whose edges carry code that fits a classical component (a warp, a Gaussian process, boosted trees, a gate) on the rows of the table at hand, whose nodes merge what the edges produce, and whose head combines the resulting predictions with non-negative weights fitted out of fold. The graph is shared across tables and frozen; its fitted state belongs to one table. Calling $f(D, X^*)$ fits every edge on D and reads off X^*, which is in-context learning by program execution.

The program was not written by hand, except for one step of its lineage, which Section 2.3 describes. TabEvo builds it with a large language model (LLM) (Romera-Paredes et al., 2024; Novikov et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

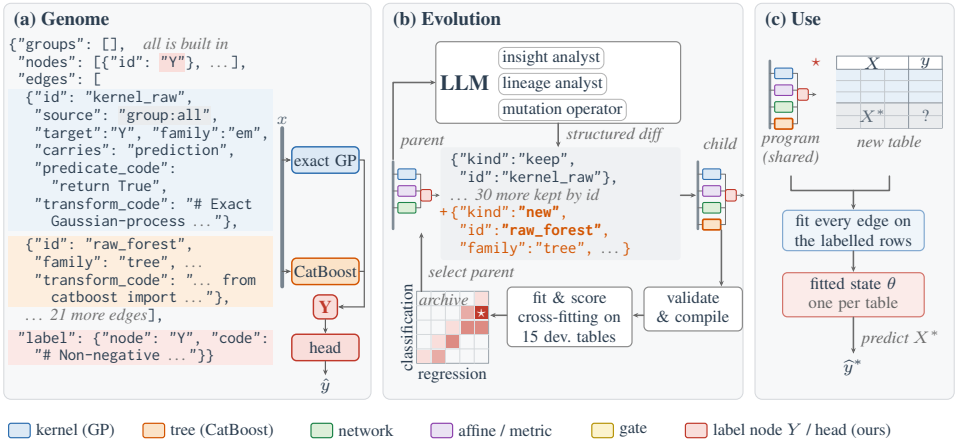


Figure 1: TabEvo at a glance. **(a) Genome.** A program is a JSON document of column groups, merge nodes and edges. The two edge records shown are the evolved program's exact Gaussian process and CatBoost lane, which read all columns and feed the label node Y; its non-negative head combines the prediction blocks. **(b) Evolution.** Three LLM roles prepare a structured diff against a parent; the child is validated, compiled, fitted with cross-fitting on the 15 development tables and archived in a MAP-Elites grid over its regression and classification scores. **(c) Use.** The frozen program ($\star$) is shared by all tables; on a new table every edge is fitted on its labelled rows, giving a table-specific state θ.

2025). A *genome* is a JSON document of column groups, merge nodes and edges with their code (Figure 1). An agent loop grows it one edit at a time, and an evolutionary search then edits it through *structured diffs* in a schema generated from the parent; each child is fitted on fifteen development tables with cross-fitting and archived by MAP-Elites (Figure 2).

We evaluate the program as a new learning algorithm, under a pre-registered protocol, on 95 tables against 45 baseline configurations (Section 4). On 22 of the 26 fresh TabArena classification tables (three exceed the time limit, one fails numerically) it is within one point, by our equivalence rule, of seven of the nine foundation models, 0.7 to 0.9 points behind TabFM and LimiX, and level with bagged CatBoost and RealMLP and with a Super Learner that stacks CatBoost, LightGBM and RealMLP under the program's head. On ten held-out tables and on 23 of 24 fresh regression tables it is ahead of every trained model, on regression ahead of the Super Learner by a small margin as well, and about a point behind the newest TabPFN versions. Section 5 measures its properties as a program. On synthetic tables it exploits repeated rows and keeps its accuracy with two thousand mostly irrelevant features where the foundation models lose theirs. Removing any single block other than the boosted lane changes its AUC by at most 0.12 points, none detectably. An automated edit-and-test loop removes half of its edges at a development score within 0.0005 of the original and makes it 6.5$\times$ faster.

Our contributions are (i) meta-DAGs, an executable representation of tabular in-context learners with cross-fitted prediction boundaries; (ii) an LLM-guided search over them, an agent loop followed by evolution with parent-specific structured diffs; (iii) a discovered program that, without pretraining, is within a point of most foundation models on fresh classification tables and level with the strongest trained models and with a Super Learner of them, in a pre-registered benchmark of 95 tables and 45 baseline configurations; and (iv) an analysis showing that an explicit learner can be stress-tested, read and simplified with ordinary tools.

2 TabEvo

2.1 Genome: a learner written as a graph

A table provides n labelled rows (X, y) and query rows X^*. A genome G is a complete description of a learner, $\theta = \mathrm{Fit}_G(X, y)$ its fitted state on this table and $\hat{Y}^* = \mathrm{Predict}_G(\theta, X^*)$ its predictions; G is fixed before any evaluation table is seen. Concretely, a genome is a JSON document with three lists, `groups`, `nodes` and `edges`, and a `label` head (Figure 1a; Appendix C.1). *Groups* select input columns. *Nodes* are ordered merge points that concatenate or multiply the blocks reaching them. *Edges* carry all the computation. Each names its `source` (a group or an earlier node) and

its `target` (a later node). Its `family` is the type of the fitted operator it returns: an affine map (`lin`), a network (`mlp`), a kernel machine (`em`) or a boosted ensemble (`tree`). What it `carries` is either a *space*, coordinates that later fitters read, or a *prediction*, a value that must be cross-fitted before anything downstream reads it. An optional `residual_edge` names the baseline subtracted from its target, and a `predicate_code` decides on the table at hand whether the edge is active. Its `transform_code` is a Python function of the source block, the labels and a view of the table (size, task type, seed); it fits whatever it likes and returns the fitted operator. Every fitter regresses the same canonical targets (standardised label or one-hot class code), so blocks of different families align column by column.

The component library. The code of an edge runs in a fixed namespace: NumPy, PyTorch, the four native operator constructors, a few helpers and any installed library; our programs import CatBoost. The blocks that occur in our programs are *exact Gaussian processes* with a powered-exponential kernel $k(x, x') = s_f^2 \exp(-(\|x - x'\|/\ell)^q)$, whose lengthscale, exponent, signal and noise variances are fitted by Adam on the exact marginal likelihood (`em`); *input warps*, a per-column Kumaraswamy warp $1-(1-u^a)^b$ with a lengthscale per column, fitted jointly with a GP by marginal likelihood, and a quantile warp to normal scores (`mlp`, carrying a space); a *linear metric* $M = Lx$ fitted by the marginal likelihood of an exponential-kernel GP (`lin`); *CatBoost* on the raw row or on a residual, depth $\approx \log_2(n/8)$, trees chosen on an internal holdout before a refit on all rows (`tree`); a three-layer *MLP* of width 256 with AdamW and early stopping (`mlp`); and three *repeat detectors*: a hard novelty score (0 on rows that repeat a training row), a soft one $1/(1 + \sum_i e^{4-d_i/s})$ from the distances d_i to the training rows in the metric, and the twin block, the mean label of the training rows identical to the query row (all `em`). A gate multiplies a novelty score into a prediction at a product node.

The label node collects B prediction blocks $f_1, \ldots, f_B$, and a non-negative stacking head combines them,

$$\widehat{y}(x) = a + \sum_{b=1}^{B} w_b \, f_b(x), \qquad (a, w) = \arg \min_{a,\, w \geq 0} \sum_{i=1}^{n} \Big\| t_i - a - \sum_b w_b \, \tilde{f}_b(x_i) \Big\|^2, \tag{1}$$

where t_i is the canonical target and $\tilde{f}_b(x_i)$ the out-of-fold prediction of block b for row i. A block the table cannot use receives weight zero rather than a signed correction; for classification the non-negative class scores are clipped and renormalised into probabilities.

Execution with honest prediction boundaries. For every edge whose prediction is read downstream, TabEvo fits the *entire upstream prefix* of that edge on $K - 1$ of K row folds and predicts the held-out fold, so no reader sees a value computed with the label of the same row. Space edges stay within the scope of their reader. The head is fitted on the out-of-fold blocks, and every edge is finally refitted on all rows.

2.2 Evolution by structured diffs

The LLM writes a diff against a parent rather than a whole genome, and the diff describes the complete child. Every group, node and edge is either `keep`, a reference to a parent element with optional replacements of single fields such as one edge's code, or `new`, a full record; whatever the diff omits is deleted. The schema of the diff is generated from the parent, so the identifiers the LLM may keep are enumerated, and a declared intent (`add_edge`, `simplify`, ...) is checked against the result. Before a child is fitted, acyclicity and every reference are verified and all code is compiled (Figure 7). Three LLM roles prepare a proposal: an *insight analyst*, working from the per-table scores of a parent and its relatives; a *lineage analyst*, working from its ancestors and children; and the *mutation operator*, which returns the diff with a justification. Children enter a MAP-Elites archive (Mouret & Clune, 2015) whose cells are the pair of mean regression and classification scores.

2.3 Search protocol

Seed. The seed program was itself produced by an agent loop. An LLM agent, given the genome format, the component library and the fitting interface, proposed one edit at a time to a minimal genome and kept the edits that improved the development score. At the start of the search the seed

had the four GP views, the residual learners and the repeat detectors, and no direct gradient-boosted lane.

Development tables and fitness. The development set is fifteen TabArena tables (eight regression, seven classification); the five largest are capped at 4,000 rows by a fixed stratified subsample. Fitness is the mean 3-fold cross-validated score (accuracy or R^2) over these tables, with $K = 3$ cross-fitting folds inside each. A failing genome receives no fitness.

Proposals and selection. The mutation operator and the two analysts ran on GPT-5.6 at high reasoning effort; summaries of past proposals were written by a second GPT-5.6 model. Archive cells are 0.002 wide in both scores and a child replaces the occupant of its cell if its fitness is higher; parents are drawn with probability $\propto \exp(-\Delta/\tau)$, Δ the gap to the best fitness and $\tau = 0.0025$, restricted to genomes with the non-negative head. The run lasted 11.7 hours and about 46 GPU-hours, with 59 proposals, 51 evaluated children (50 valid), 407 LLM calls and 18.6 M tokens. One step of the selected program's lineage was ours rather than the LLM's. A check of the best ridge-headed genome (fitness 0.79254) on the ten held-out tables showed its head collapsing on qsar-tid-11 (R^2 0.007 against the seed's 0.72). We therefore restricted parents to the non-negative head, replaced that genome's head by the seed's stacking head and submitted the result as a child, which was evaluated like every other (0.78929; Table 18). On the fresh tables, that ridge-headed parent is indistinguishable from the selected program on 19 of the 22 tables that both complete (hiva-agnostic fails for both) and collapses on bioresponse (-26.7 points, parent minus program), anneal (-23.0) and mic (-4.8). Selected program minus parent is $+2.36$ [$+0.03$, $+5.56$] on 23 tables, with hiva-agnostic (both fail) counted as a tie (W/T/L 9/8/6); parent minus seed is -2.42 [-5.70, $+0.01$] on the 22 tables that both complete. The search's fitness rewarded a head that transfers worse, and the selected program is the product of the manual head choice rather than of the search. The two best archive members were scored on the same ten held-out tables and the one with the higher mean was frozen, so those ten tables took part in choosing the program.

3 THE EVOLVED PROGRAM

Read as an algorithm. The selected program (Figure 2) is a Gaussian-process method that does not commit to one geometry of the input space. Four exact GPs predict in four coordinate systems, and the metric GP also serves as the baseline of two residual learners. Gates and the twin block handle exact repetition, and the last block, the edge the search added, is a CatBoost lane on the raw row. Every edge of the selected program is unconditionally active, since the search never used the predicates; the program's dependence on the dataset therefore lies entirely in its fitted state.

4 BENCHMARK

Setup. We evaluate the frozen program the way a tabular foundation model is evaluated: one fit on the training rows of a table it has never seen, one prediction on the test split, and no per-table tuning. The protocol was written down before any baseline result was read, and each later addition before its own results existed (Appendix A). Three groups of tables carry the comparison (Table 1). The *fresh classification* group is the 26 classification tasks of TabArena v0.1 (Erickson et al., 2025), TabArena-Lite split, which played no part in the development of the program. The *held-out* group is ten tables, five of them regression, that the search consulted twice, once to check a candidate and once to choose the final program (Section 2.3); they are close to fresh but not fully independent of it. The *fresh regression* group is 24 regression tables from the TabM, TabReD and Grinsztajn benchmark suites, none of them used anywhere in development; it holds every regression table in our data store that belongs to no other group, less the five that duplicate a TabArena dataset. Two further groups are reported in the appendix: the 15 development tables, in-sample for the program, and 20 high-dimensional OpenML tables, where we expected a growing margin and found none (Appendix H.4). Foundation models run at their official inference settings. Trained baselines use library defaults and the meta-tuned defaults of Holzmüller et al. (2024), the strongest of them bagged over five folds (b5). The program fits its exact GPs on at most 10^4 training rows. A method that does not finish a table within two hours (one hour on the regression group) has no score there, and the table is left out of its comparisons. TabEvo does not complete four of the 26 fresh classification tables and one

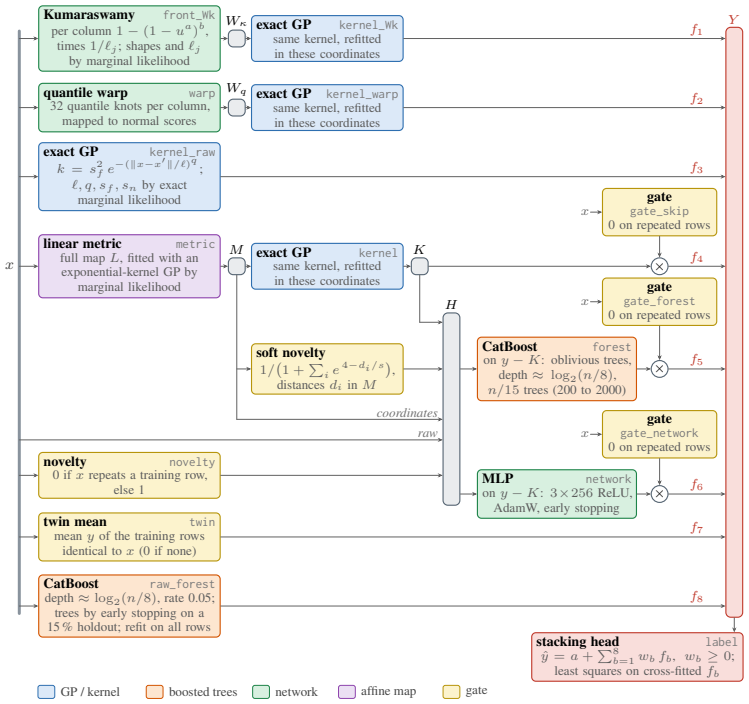


Figure 2: **The evolved program.** Boxes are fitted operators, filled by family (legend); the grey tag is the edge's id and the grey line its key setting. Plain arrows pass a block through, rounded grey nodes concatenate, $\otimes$ multiplies, x is the input row and n the number of training rows. Three exact GPs predict from the Kumaraswamy-warped row W_κ, the quantile-warped row W_q and the raw row; a fourth works in the learned metric $M = Lx$, and its prediction K is both the block f_4 and the baseline of a CatBoost and an MLP fitted to $y - K$ on $H = [x, M, K, \text{novelty}, \text{soft novelty}]$. Gates zero K and both residual predictions on query rows that repeat a training row, where the twin block returns the mean label of the identical rows; a second CatBoost (`raw_forest`, the edge the search added) reads x directly. The head combines the eight blocks (Eq. 1).

of the 24 regression tables. On amazon-employee-access, kddcup09-appetency and sdss17, one-hot encoding produces 5,000 to 28,000 columns and the exact GPs run past the limit; on hiva-agnostic a Cholesky factorisation fails; on microsoft the fit is degenerate, with R^2 below zero, and we count it as a failure by a rule adopted when the case arose. Appendix I.4 repeats every analysis with these tables counted as the worst result; under that rule the program trails the foundation models by 0.9 to 1.5 points and Stack-TD by 1.1 on classification, and is level with Stack-TD on regression.

Statistics. Everything is paired per table. Δ is TabEvo minus the method, in points of accuracy or R^2, with a 95 % bootstrap interval over tables; a difference counts as clear when that interval excludes zero. An interval that includes zero does not show equality, so we also use an equivalence rule: two methods count as level only when the whole interval lies within one point. Wilcoxon signed-rank tests with Holm correction, a stricter standard, win/tie/loss counts and every per-table score are in Appendix I.

Baselines. We compare against 40 methods in 45 configurations (Appendix B): nine foundation models (TabFM, TabPFN v3, v2.5 and v2, TabICLv2 and v1.1, Mitra, TabDPT, LimiX; Kong & Das, 2026; Grinsztajn et al., 2026; 2025; Hollmann et al., 2025; Qu et al., 2026; 2025; Zhang et al., 2025b; Ma et al., 2025; Zhang et al., 2025a; 2026); CatBoost, XGBoost and LightGBM (Prokhorenkova et al., 2018; Chen & Guestrin, 2016; Ke et al., 2017) and other tree ensembles; sixteen deep tabular models (TabM, RealMLP, TabR, FT-Transformer, . . . ; Gorishniy et al., 2025; Holzmüller et al., 2024; Gorishniy et al., 2024; 2021); and classical methods, among them an exact GP with a lengthscale per feature and xRFM (Beaglehole et al., 2026). Stack-TD is a Super Learner (van der Laan et al., 2007). It fits CatBoost, LightGBM and RealMLP with their tuned defaults, obtains their out-of-fold predictions by three-fold cross-fitting and combines them with the program's own non-negative stacking head. It isolates what the head contributes when it sits on standard learners instead of the discovered graph, and it is the comparison we hold the program to most strictly.

Table 1: **Results.** Mean score ($\times 100$: accuracy, or R^2 for regression) on the tables TabEvo completes in each group, and Δ = TabEvo − method in points, both with 95 % bootstrap intervals over tables in small type. Shading of Δ: grey, level by the equivalence rule (interval within one point); otherwise green, TabEvo clearly better (interval above zero); red, the method clearly better (interval below zero). Bold: best of the column; a method that misses tables is paired on the tables both complete; b5: bagged over five folds; TD/D: tuned/library defaults; –: not run. Regression: 4,000 training rows, test split capped at 20,000 rows for every method. Stack-TD: Super Learner of CatBoost-TD, LightGBM-TD and RealMLP-TD with TabEvo's head.

	Fresh TabArena classification (22 tables)		Held-out (10 tables)		Fresh regression (23 tables)	
Method	Score	$\Delta_{[95\%\ \text{CI}]}$	Score	$\Delta_{[95\%\ \text{CI}]}$	Score	$\Delta_{[95\%\ \text{CI}]}$
TabEvo	88.3$_{[84.6, 91.8]}$		82.5$_{[69.8, 91.6]}$		71.3$_{[61.0, 81.0]}$	
Foundation models						
TabFM	**89.0**	−0.66$_{[-1.03, -0.31]}$	84.1	−1.64$_{[-3.04, -0.39]}$	**72.6**	−1.35$_{[-2.45, -0.42]}$
TabPFN v3	88.7	−0.42$_{[-0.73, -0.13]}$	83.5	−0.97$_{[-1.84, -0.03]}$	72.4	−1.12$_{[-1.87, -0.49]}$
TabPFN v2.5	88.7	−0.36$_{[-0.66, -0.06]}$	82.9	−0.39$_{[-1.08, 0.32]}$	72.1	−0.82$_{[-1.43, -0.29]}$
TabICLv2	88.6	−0.29$_{[-0.62, 0.04]}$	82.6	−0.08$_{[-1.01, 0.97]}$	70.6	+0.71$_{[-0.82, 2.95]}$
TabDPT	88.6	−0.23$_{[-0.60, 0.17]}$	83.2	−0.67$_{[-1.15, -0.22]}$	–	
LimiX	88.5	−0.85$_{[-1.22, -0.48]}$	**84.8**	−2.28$_{[-3.54, -1.21]}$	–	
TabPFN v2	88.4	−0.03$_{[-0.42, 0.43]}$	82.0	+0.46$_{[-0.61, 1.69]}$	71.3	−0.08$_{[-0.72, 0.68]}$
Mitra	88.0	−0.20$_{[-0.66, 0.26]}$	88.7	+0.23$_{[-0.65, 1.32]}$	–	
Gradient boosting and stacking						
Stack-TD	88.5	−0.17$_{[-0.56, 0.15]}$	82.3	+0.19$_{[-0.78, 1.02]}$	70.6	+0.64$_{[0.09, 1.26]}$
CatBoost-TD$_{\text{b5}}$	88.3	+0.01$_{[-0.26, 0.24]}$	82.2	+0.25$_{[-0.86, 1.21]}$	69.6	+1.67$_{[0.76, 2.68]}$
CatBoost-D	88.1	+0.22$_{[-0.07, 0.53]}$	81.2	+1.34$_{[-0.30, 3.06]}$	69.6	+1.70$_{[0.82, 2.74]}$
Ensemble-TD	88.1	+0.25$_{[-0.12, 0.61]}$	81.7	+0.84$_{[-0.52, 2.40]}$	70.5	+0.76$_{[0.12, 1.46]}$
LightGBM-TD$_{\text{b5}}$	88.0	+0.32$_{[0.02, 0.62]}$	81.9	+0.61$_{[-0.14, 1.43]}$	69.6	+1.66$_{[1.05, 2.38]}$
LightGBM-D	87.8	+0.50$_{[0.15, 0.91]}$	80.9	+1.55$_{[0.39, 2.80]}$	69.0	+2.29$_{[1.43, 3.25]}$
XGBoost-TD$_{\text{b5}}$	87.8	+0.53$_{[0.16, 0.93]}$	81.0	+1.49$_{[0.16, 3.41]}$	69.1	+2.20$_{[1.45, 3.00]}$
XGBoost-D	87.8	+0.54$_{[0.19, 0.92]}$	79.7	+2.82$_{[0.91, 5.08]}$	66.6	+4.64$_{[3.16, 6.17]}$
Random forest	87.6	+0.75$_{[0.38, 1.13]}$	80.6	+1.89$_{[0.90, 3.05]}$	67.2	+4.02$_{[2.59, 5.55]}$
Neural networks						
RealMLP-TD$_{\text{b5}}$	88.3	+0.02$_{[-0.23, 0.26]}$	81.4	+1.12$_{[0.14, 2.29]}$	70.4	+0.85$_{[-0.13, 2.09]}$
FT-Transformer-D	88.2	+0.51$_{[0.07, 1.03]}$	79.9	+2.62$_{[0.70, 5.36]}$	67.6	+3.26$_{[1.58, 5.24]}$
TabM-D$_{\text{b5}}$	88.0	+0.31$_{[-0.05, 0.68]}$	81.1	+1.37$_{[0.24, 2.85]}$	69.9	+1.39$_{[0.35, 2.66]}$
TabR-D	87.6	+0.74$_{[0.22, 1.27]}$	81.4	+1.11$_{[0.49, 1.76]}$	68.4	+2.90$_{[1.81, 4.09]}$
RealTabR-D	87.6	+0.76$_{[0.25, 1.31]}$	81.2	+1.32$_{[0.38, 2.63]}$	70.0	+1.30$_{[0.68, 1.96]}$
MLP-PLR-D	87.4	+0.94$_{[0.24, 1.75]}$	79.9	+2.55$_{[1.02, 4.27]}$	69.0	+2.26$_{[1.05, 3.85]}$
MLP-D	87.4	+0.94$_{[0.29, 1.65]}$	80.7	+1.84$_{[0.87, 2.82]}$	67.1	+4.15$_{[2.34, 6.55]}$
ResNet-D	87.4	+0.96$_{[0.37, 1.63]}$	80.6	+1.92$_{[0.89, 3.09]}$	67.2	+4.03$_{[2.47, 5.97]}$
Classical						
GP (ARD, exact)	87.5	+0.79$_{[0.31, 1.31]}$	80.2	+2.32$_{[0.70, 4.49]}$	67.7	+3.54$_{[1.10, 6.74]}$

Against foundation models. On the fresh classification tables the program is level, by the equivalence rule, with seven of the nine foundation models: TabICLv2, TabPFN v3, v2.5 and v2, TabDPT and Mitra (Table 1; Mitra on the 16 tables it completes), and TabICL v1.1 (Table 58). Each of the seven wins more tables than it loses and leads by a fraction of a point on average. TabPFN v3 and v2.5 are ahead by a few tenths of a point, intervals excluding zero but within the margin. Two are ahead by more than the margin allows: TabFM, whose developers report evaluating it on these TabArena tables ($\Delta = -0.66$ points $[-1.03, -0.31]$), and LimiX (-0.85 $[-1.22, -0.48]$). A critical-difference analysis over the eighteen strongest methods separates the program from those two only (Appendix I.2). The held-out tables repeat the picture with a wider spread: the program is within a point of TabICLv2, TabPFN v2.5, v2 and TabDPT on average, and 1.0 to 2.3 points behind TabPFN v3, TabFM and LimiX. On the fresh regression tables it is level with TabPFN v2 (-0.08 $[-0.72, +0.68]$). It is ahead of TabICLv2 by $+0.71$ $[-0.82, +2.95]$, an interval that includes zero, and clearly behind TabPFN v3 (-1.12 $[-1.87, -0.49]$), TabPFN v2.5 (-0.82 $[-1.43, -0.29]$) and TabFM (-1.35 $[-2.45, -0.42]$).

Against trained models and the Super Learner. With no pretraining, the program is level with the strongest trained baselines (bagged CatBoost, RealMLP and TabM) on the fresh classification tables and ahead of every other trained configuration. On the held-out tables it is ahead of every trained model, most of them clearly. Stack-TD, the Super Learner of three of them under the pro-

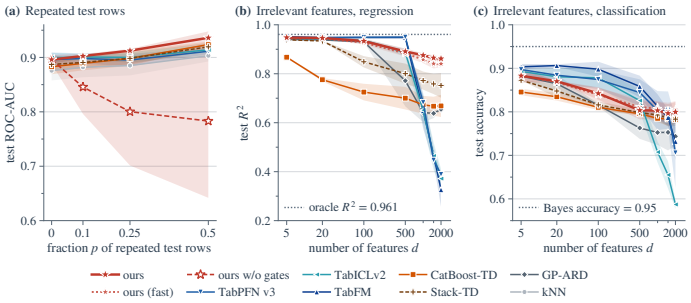


Figure 3: **Controlled experiments**; means and one-standard-deviation bands over three data seeds. (a) Repeated rows: a noisy binary task in which a fraction p of the test rows copies training rows, features and noisy labels alike. (b, c) Irrelevant features: Friedman's function of five informative features plus $d - 5$ irrelevant Gaussian ones, the 500 training rows and their labels fixed as d grows (dotted: oracle R^2 and Bayes accuracy); "ours (fast)" is the pruned program of Section 5.3, Stack-TD the Super Learner of Section 4.

gram's head, lands where the program lands: -0.17 points $[-0.56, +0.15]$ on the fresh tables (level) and $+0.19$ $[-0.78, +1.02]$ on the held-out tables. On the fresh regression tables the program is ahead of every trained model, clearly so for all but bagged RealMLP, and by a smaller margin ahead of the Super Learner ($\Delta = +0.64$ $[+0.09, +1.26]$, W/T/L 17/0/6). An exact GP with a lengthscale per feature is on its own far behind the program ($\Delta = +3.54$ $[+1.10, +6.74]$); adding that GP to the Super Learner's base learners narrows the gap, on the 17 tables where that stack completes, to $\Delta = +0.24$ $[-0.28, +0.63]$, level by the equivalence rule. With the regression table the program does not complete counted against it, program and stack are level, and the margin we had set in advance for a practical advantage over the stack is not reached (Appendix I.4). On the synthetic tables of Section 5 the program is ahead of the stack at every point. Against its own seed the program gains $+0.06$ points $[-0.05, +0.17]$ on the fresh tables (Appendix I); the interval includes zero, and the evolutionary stage contributed one lane.

Table size. The number of rows explains most of the spread among the strongest methods. On the 16 fresh and held-out tables with at most 4,000 development rows, where the program fits every row with exact GPs, it is level with TabPFN v2, within a point of TabICLv2, and beats bagged CatBoost on 11 of the 16. On the larger tables, where it fits a 10^4-row subsample and the baselines see every row, TabICLv2 and TabPFN v2.5 lead by half a point and LimiX by one (Table 64; learning curves in Figure 4a).

5 Analysis: The Program as an Object of Study

5.1 Controlled Experiments

Repeated rows. When a growing fraction of the test rows copies training rows (Figure 3a), the program's AUC rises with the fraction and is the highest at every $p > 0$. The gates make this work. On repeated rows they silence the residual learners, which are least reliable at their own training points, and leave the GPs, which interpolate the training labels. Without the gates the program falls below every baseline.

Irrelevant features. With the five informative features and the labels fixed, we add irrelevant ones (Figure 3b, c). The foundation models lead by two points at $d = 5$. Beyond a thousand features their regression accuracy collapses, and at two thousand their classification accuracy follows. The program loses 8.6 points of R^2 from five to two thousand features and is then above every baseline on every metric. Its R^2 is 0.86, against 0.67 for CatBoost, 0.75 for the Super Learner and 0.33–0.39 for the foundation models (TabFM reads at most 500 features, so part of its loss is truncation); its accuracy is 0.80, 1.7 points above the stack and 6.8 above the best foundation model. The Super Learner lies below the program at every point of all three experiments: by 8.5 to 10.9 R^2 points beyond a hundred features, and by small margins within seed noise on classification and on repeated rows. The program's resistance to the added features comes from the composition, since an exact GP with a lengthscale per feature loses 28.6 points of R^2 over the same range. The Kumaraswamy warp's per-column lengthscales rank the five informative features above the rest (Appendix H.5.1).

Table 2: **Ablations** on the fresh tables in the 4,000-row regime (larger tables subsampled; no time limit): ΔAUC = variant − program in points with its 95 % bootstrap interval, wins/losses of the variant, Wilcoxon p (Holm over rows), Δlog-loss, and the share of development folds in which the head gave the removed block weight zero; n: tables on which both completed.

Removed block	n	ΔAUC [95% CI]	W/L	p	p_{Holm}	Δlog-loss	zero wt.
Leave one block out							
GP on Kumaraswamy warp	8	−0.09 [−0.31, 0.08]	2/4	.500	1.000	+0.003	20%
GP on quantile-normal warp	8	+0.10 [−0.11, 0.37]	3/1	.500	1.000	−0.001	29%
GP on raw features	8	+0.09 [−0.06, 0.30]	2/2	.750	1.000	−0.001	51%
metric-GP lane	8	+0.12 [−0.01, 0.33]	5/2	.219	1.000	0.000	11%
residual CatBoost lane	22	−0.03 [−0.17, 0.07]	12/9	.721	1.000	+0.002	29%
residual MLP lane	22	+0.05 [−0.01, 0.15]	7/6	.695	1.000	0.000	33%
twin-mean lane	23	−0.02 [−0.05, 0.01]	4/4	.900	1.000	0.000	44%
direct CatBoost lane†	25	−0.41 [−0.74, −0.12]	6/17	.024	.262	−0.003	4%
Other changes and single-family references							
novelty gates (3)	22	+0.05 [0.00, 0.15]	5/6	.961	1.000	0.000	–
all non-GP lanes (GP only)	25	−0.53 [−0.98, −0.15]	6/19	.010	.116	0.000	–
all but direct CatBoost	25	+0.04 [−0.55, 0.83]	10/15	.300	1.000	+0.031	–
NNLS head → ridge head	8	−0.22 [−0.64, 0.05]	5/3	1.000	1.000	+0.103	–

† On 17 of these tables the cell is taken from the main-benchmark run of the same genome; where both exist they are identical. zero wt.: share of the 45 development folds (15 tables × 3) in which the NNLS head gave the removed block weight exactly 0 (– = not a single block).

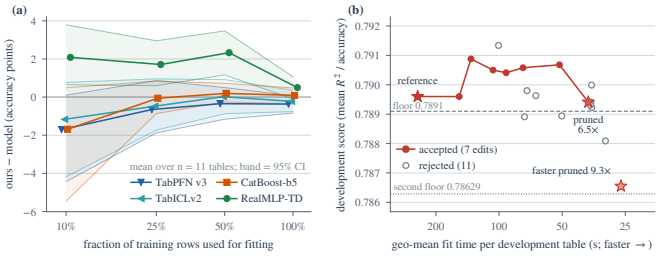


Figure 4: **Data and speed.** (a) TabEvo minus each model in accuracy points (mean and 95 % bootstrap interval over the eleven fresh tables with at most 4,000 rows; a failed fit scores as the worst model) when every method is refitted on a fraction of the training rows. (b) The edit-and-test loop: development score against fit time for the 19 logged programs; filled points and the line are the accepted chain, open points rejected edits, dashed the acceptance floor, dotted the looser floor of a second, faster pruned program (Appendix F).

5.2 What the program relies on

Ablations (Table 2) agree with the per-table head weights (Figure 19). Removing any one GP view, either residual learner, the twin block or the gates, and refitting the head on the rest, moves the AUC on the fresh tables by at most 0.12 points (none significant; $n = 8$ to 23, differing by row). No single block is measurably necessary, and the head switches each of them off on 11 to 51 % of the development folds. The one block whose removal costs something is the CatBoost lane that the search added (−0.41 points [−0.74, −0.12], worse on 17 of 25 tables, $p = .024$, Holm .26). Removing a whole family costs more. The four GP views on their own lose half a point of AUC and lose on 19 of 25 tables ($p = .010$, Holm .12); the CatBoost lane on its own keeps the AUC but worsens the log-loss by 0.03; and an unconstrained ridge head in place of the non-negative one moves the AUC by −0.22 [−0.64, 0.05] ($n = 8$) and degrades the log-loss by 0.1.

5.3 Editing the program

An edit-and-test loop. An LLM agent profiled the fitting time, proposed one change at a time, re-scored the program on the development tables, and kept the change only if the score stayed within 0.0005 of the reference and the time fell by at least a tenth (acceptance rule and log in Appendix F). Seven of eighteen edits were kept (Figure 4b): reused cross-fitted pass-throughs, subsampled hyper-parameter searches, cheaper boosted lanes, no residual lanes, and two cross-fitting folds. The pruned program has 12 edges instead of 23 and fits 6.5× faster on the development tables at an unchanged

development score (0.7894 vs. 0.7896 re-scored). On the fresh tables its ROC-AUC and log-loss do not change measurably and its accuracy drops by 0.3 points (Appendix F).

From a single GP to the program. A from-scratch loop (Table 20, Appendix D) places the evolved program $+2.2$ points above a single exact GP with the same head ([0.8, 3.8], up on 11 of 15 development tables, $p = .004$), mostly on regression ($+3.9$).

6 Related work

Prior-data fitted networks are pretrained on synthetic tasks to predict from a labelled context in one forward pass (Müller et al., 2022; Hollmann et al., 2023). Their successors add regression and scale (Hollmann et al., 2025; Grinsztajn et al., 2025; 2026; Qu et al., 2025; 2026; Zhang et al., 2025b;a; 2026; Kong & Das, 2026) or pretrain on real tables (Ma et al., 2025). TabEvo keeps their interface but replaces pretraining by program search. The Super Learner (van der Laan et al., 2007) and AutoGluon (Erickson et al., 2020) stack a fixed library of learners with weights fitted anew on each dataset; TabRepo (Salinas & Erickson, 2024) learns portfolios of configurations across datasets; TPOT evolves a pipeline per dataset (Olson et al., 2016). An exploratory LLM search over whole Python fit–predict functions (Appendix J) yields programs that complete every table but trail the meta-DAG program on every common slice, in a different setting (768 context rows and a synthetic-prior objective). TabEvo searches across datasets for the composition itself, code included, and fixes it before a dataset is seen, so that only the components and the head are refitted. AutoML-Zero evolved learning algorithms from primitives (Real et al., 2020), and LLMs now act as program mutation operators (Lehman et al., 2023; Romera-Paredes et al., 2024; Novikov et al., 2025) with MAP-Elites archives (Mouret & Clune, 2015); the search runs on the GigaEvo framework (Khrulkov et al., 2025). The components are classical: exact GPs (Rasmussen & Williams, 2006), input warping (Snoek et al., 2014), learned kernel metrics and non-negative stacking (van der Laan et al., 2007) on cross-fitted predictions (Chernozhukov et al., 2018).

These approaches adapt at different levels. A prior-data fitted network conditions predictions on the supplied context, whereas AutoML selects a configuration for each table. Here the outer search transfers one executable learning rule across tables, while inner fitting estimates metrics, predictors, and combination weights from each new context. This distinction makes the transferred procedure, not only its predictions or selected components, an object of comparison and analysis.

7 Limitations and conclusion

Limitations and outlook. The program trails most foundation models by a few tenths of a point on the fresh tables it completes, trails TabFM and LimiX by 0.7 to 0.9, and has no result on four of the 26 (Table 1; Appendix I counts them against it). A Super Learner of three trained baselines with the same head matches its classification accuracy where the program runs and leads by 1.1 points when the tables the program does not complete count against it, so on classification the real-table accuracy is not specific to the discovered program. Its advantage over that stack is confined to the regression tables it completes and to the synthetic tests. Under a covariate shift along the first principal component it loses more accuracy than the foundation models (Appendix H.2), and the margin we had expected to grow on high-dimensional tables does not appear on real ones. The search was one run, and its fitness signal rewarded a head that transfers worse (Section 2.3).

Conclusion. A tabular in-context learner need not be a neural network. TabEvo implements the same function as an explicit program, a meta-DAG of classical components with cross-fitted prediction boundaries, found by LLM-guided search and frozen before any fresh table is seen. On the fresh tables it completes, the discovered program is within one point of seven of nine foundation models. It is level with the strongest trained models and with a Super Learner of them on classification, and ahead of every trained model on the regression tables it completes, where its margin over the Super Learner is small and disappears once a Gaussian process joins the stack. It holds up on synthetic tables where the foundation models degrade. It can be read, ablated and edited; an automated loop halved it at an unchanged development score. It is under a point behind TabFM and LimiX, it fails on one-hot-heavy tables, and it gained little from the evolutionary stage.

ETHICS STATEMENT

This work uses only public benchmark tables (TabArena v0.1, OpenML, and the TabM, TabReD and Grinsztajn regression suites), under their published licences and splits. No data were collected, no human subjects or annotators were involved, and no attempt was made to identify individuals in any table. The method searches over programs that fit classical statistical components and introduces no capability beyond tabular prediction. Its main foreseeable risk is the usual one of automated model search: over-claiming from a search that has seen its evaluation data. We address it by keeping the fresh tables out of development, by reporting the failures of the program alongside its results, and by releasing every record behind every number. The search consumed about 46 GPU-hours and 18.6 M LLM tokens (Section 2.3).

REPRODUCIBILITY STATEMENT

The genome of the discovered program, the search configuration, the benchmark protocol with its dated changes, the list of all 95 tables with their splits, the runners for every method and the per-cell result records will be released with the paper. Section 2 and Appendix D define the program representation, the fitting protocol and the search; Appendices A and B define the evaluation.

LLM USAGE STATEMENT

Large language models are part of the method. During the search they propose the program edits and the analyses of parents and lineages that motivate them (Section 2.2; models and call counts in Appendix D), and an LLM agent loop built the seed program (Section 2.3). Beyond the method, LLM-based coding assistants were used to implement the benchmark runners, the data-preparation and analysis scripts and the pruning loop of Section 5.3, to run experiments under the authors' instructions, to help design the statistical analysis, to draft and edit the text and the figures, and to summarise results. Every number in the paper is generated from the released result records by the released scripts, which the authors reviewed. The authors designed the study, made every decision recorded in the protocol and take full responsibility for the content.

REFERENCES

Daniel Beaglehole, David Holzmüller, Adityanarayanan Radhakrishnan, and Mikhail Belkin. xRFM: Accurate, scalable, and interpretable feature learning models for tabular data. In *International Conference on Learning Representations*, 2026.

Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *Journal of Machine Learning Research*, 17(5):1–10, 2016.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. doi: 10.1145/2939672.2939785.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018. doi: 10.1111/ectj.12097.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-Tabular: Robust and accurate AutoML for structured data. *arXiv preprint arXiv:2003.06505*, 2020. URL `https://arxiv.org/abs/2003.06505`.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 38, 2025.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18932–18943, 2021.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24991–25004, 2022.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. TabR: Tabular deep learning meets nearest neighbors. In *International Conference on Learning Representations*, 2024.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025. URL https://arxiv.org/abs/2511.08667.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026. URL https://arxiv.org/abs/2605.13986.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025. doi: 10.1038/s41586-024-08328-6.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. In *Advances in Neural Information Processing Systems*, volume 37, pp. 26577–26658, 2024.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Valentin Khrulkov, Andrey Galichin, Denis Bashkirov, Dmitry Vinichenko, Oleg Travkin, Roman Alferov, Andrey Kuznetsov, and Ivan Oseledets. GigaEvo: An open source optimization framework powered by LLMs and evolution algorithms. *arXiv preprint arXiv:2511.17592*, 2025. URL https://arxiv.org/abs/2511.17592.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research Blog, June 2026. URL https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/.

P. Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1–2):79–88, 1980. doi: 10.1016/0022-1694(80)90036-0.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. In Wolfgang Banzhaf, Penousal Machado, and Mengjie Zhang (eds.), *Handbook of Evolutionary Machine Learning*, Genetic and Evolutionary Computation, pp. 331–366. Springer Nature Singapore, 2023. doi: 10.1007/978-981-99-3814-8_11.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems*, volume 38, pp. 172692–172722, 2025.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015. URL https://arxiv.org/abs/1504.04909.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. URL https://arxiv.org/abs/2506.13131.

Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 485–492. ACM, 2016. doi: 10.1145/2908812.2908918.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847. PMLR, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2602.11139.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. doi: 10.7551/mitpress/3206.001.0001.

Esteban Real, Chen Liang, David R. So, and Quoc V. Le. AutoML-Zero: Evolving machine learning algorithms from scratch. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8007–8019. PMLR, 2020.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

David Salinas and Nick Erickson. TabRepo: A large scale repository of tabular model evaluations and its AutoML applications. In *Proceedings of the Third International Conference on Automated Machine Learning*, volume 256 of *Proceedings of Machine Learning Research*, pp. 19/1–30. PMLR, 2024.

Jasper Snoek, Kevin Swersky, Richard S. Zemel, and Ryan P. Adams. Input warping for Bayesian optimization of non-stationary functions. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1674–1682. PMLR, 2014.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 567–574. PMLR, 2009.

Mark J. van der Laan, Eric C. Polley, and Alan E. Hubbard. Super learner. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007. doi: 10.2202/1544-6115.1309.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. doi: 10.2307/3001968.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025a. URL https://arxiv.org/abs/2509.03505.

Xingxuan Zhang, Gang Ren, Hao Yuan, Hao Zou, Hongze Tan, Hui Wang, Jianhao Song, Jiansheng Li, Jiayao Zhang, Jinghan Zhang, Kaifang Li, Lang Mo, Li Mao, Mingchao Hao, Nuo Xu, Rui Ding, Ruiji Zhang, Shuyang Li, Siyu Mei, Tianyang Zhang, Weiyang Mu, Yancheng Dong, Yongxian Wei, Yuan Xue, Yuanrui Wang, Yue He, Zijia Yang, Ziyun Li, Dongzhe Li, Fuqiang Wang, Jiandong Liu, Jiawei Chen, Jiaxin Du, Kaijie Cheng, Kehan Li, Lei Sun, Linjun Zhou, Ningbo Dai, Qi Wang, Renzhe Xu, Shaoxing Du, Shumeng Yang, Wang Lu, Wenjing Chu, Xiannan Huang, Xiaoyu Lin, Xing Ai, Xinyan Han, Xuanyue Li, Xuanyue Su, Xukun Zhang, Yan Lu, Yaxin Zhang, Yi Qin, Yifei Huang, Yihan Xu, Yongle Lv, Yuanyuan Jiang, Yushan Han, and Peng Cui. LimiX-2: A contextual mechanism network towards general structured-data intelligence. *arXiv preprint arXiv:2609.17488*, 2026. URL https://arxiv.org/abs/2609.17488.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 15795–15840, 2025b.

APPENDIX

Contents.

A	Benchmark protocol: data, groups, regimes, fitting rules, statistics, amendments	p. 14
B	Baseline configurations	p. 18
C	The evolved program in full	p. 21
D	The evolution run	p. 26
E	Ablations	p. 32
F	Pruning for speed	p. 34
G	Controlled toy experiments A–D	p. 38
H	Additional analyses	p. 44
I	Complete results, failures, runtime and cost	p. 55
J	Unstructured Python-program search	p. 77

Conventions. *Scores.* The primary metric is accuracy (classification) or R^2 (regression); differences are in points ($\times 100$). *Groups. Fresh* (26 TabArena tables never used in our development), *held-out* (10; consulted twice by the search), *battery* (15; the search's fitness tables, in-sample for our architecture), *high-dim.* (20 OpenML tables of a pre-registered hypothesis) and *fresh regression* (24; Amendment 17) are never pooled silently. *Model names* (the identifier in parentheses labels rows in some tables). TabEvo as reported in the main text (`ours_kvw_battery`) is the evolved program exactly as selected (at most 10,000 fit rows); the full-data variant (`ours_full`) is designated by Amendments 7, 13 and 13b, has results on 10 of the 26 fresh tables and appears only in this appendix; the seed (`ours_seed_kvw`) is the agent-designed program (built by an LLM agent loop, not by the evolutionary search) that TabEvo was evolved from. *Status of an analysis.* Every analysis is marked either *pre-registered* (written into the protocol or a dated plan before any of its results existed) or *exploratory* (decided after results were seen). *Sources.* Every number is computed from the released result records.

A BENCHMARK PROTOCOL

The protocol was written down before any baseline result was read, and every later change was added as a dated amendment before the results it governs existed (§A.5). This section restates the binding rules; the full specification and a machine-readable list of the tables are released with the code.

A.1 TABLES, SPLITS AND GROUPS

TabArena. All 51 tasks of TabArena v0.1 (Erickson et al., 2025): 38 classification tasks with at most 10 classes (30 binary, 8 multiclass) and 13 regression tasks. Each table has one split, the TabArena-Lite split (repeat 0, fold 0 of the OpenML task): the test set is fold 0 and the development (dev) rows are folds 1 and 2. We store the dev rows as train + validation, but every model receives their union. The tables have 498 to 100,000 dev rows; 25 have at most 4,000, 11 have 4,001–10,000 and 15 have more than 10,000.

High-dimensional tables. Twenty OpenML datasets with 122 to 10,935 features, fixed by name in Amendment 2 before any of their data were built: madelon, gina-agnostic, internet-advertisements, christine, dilbert, gisette, arcene, cnae-9, semeion, mfeat-factors, three gene-expression tables (ap-colon-kidney, ap-breast-ovary, ova-lung), santander-transaction-value, tecator, communities-and-crime, topo-2-1 and the three QSAR-TID regression datasets with the most rows other than QSAR-TID-11. A dataset would have been dropped, never replaced, if it were unavailable, had more than 10 classes, or appeared anywhere in our earlier development; none was dropped. Each has one random split, `train_test_split(test_size=0.2, random_state=0)` of scikit-learn on the OpenML row order, stratified on classification: dev 80 %, test 20 %.

Groups. Tables are reported in five groups that are never pooled without saying so.

- *Battery* (15 tables: 4 binary, 3 multiclass, 8 regression) are the fitness tables of the evolutionary search, so they are in-sample for the architecture. During the search the five largest were used as fixed 4,000-row subsamples.
- *Held-out* (10 tables: 4 binary, 1 multiclass, 5 regression) never entered a fitness value or an LLM prompt; they were consulted once during the search (Section 2.2) and once to choose between two finalists under a rule written before any held-out score was read (Appendix D.5).
- *Fresh* (26 tables: 22 binary, 4 multiclass) were never used in any of our work before the benchmark. The fresh group carries the out-of-sample verdict; it contains no regression table.
- *High-dim.* (20 tables: 9 binary, 4 multiclass, 7 regression) is the pre-registered test set of hypothesis H1 (§A.4).
- *Fresh regression* (24 regression tables in no other group; Amendment 17) is evaluated in the 4,000-row regime only.

Tables 3 and 4 list every table with its group, task, sizes and OpenML identifier.

A.2 Data regimes

Full data (main). Every model is fitted on *all* dev rows (up to 100,000) and predicts the full test split (Amendment 1). The 25 TabArena tables with at most 4,000 dev rows are byte-identical in both regimes, so their cells serve both analyses.

4,000-row cap (secondary). The dev rows of the 26 larger TabArena tables are capped at 4,000 by a subsample stratified by class on classification (uniform on regression), seed 0; the test split is unchanged. The cap applies to *every* method, which isolates the small-data regime the search targeted.

Rows seen by our models. The full-data variant, designated by Amendment 7 to represent TabEvo in the full-data tables (it has results on 10 of the 26 fresh tables, so the main text reports the evolved program as selected), fits on all dev rows; above 8,000 fit rows its four GP prediction edges use a subset-of-regressors/DTC solve with 8,000 inducing rows (Quiñonero-Candela & Rasmussen, 2005; Titsias, 2009), and below that the exact solve (Amendment 7; 18 TabArena tables have more than 8,000 dev rows). The evolved program as selected, TabEvo in the main text, fits exact GPs on a stratified subsample of at most 10,000 dev rows (seed 0, recorded per cell); the SoR variant uses the same approximation with 2,000 inducing rows and no row cap (Amendment 6). Baselines are never subsampled; a foundation model with a documented row or feature limit uses its official mechanism for larger data, recorded per cell (Appendix B.4).

A.3 Fitting rules

The same information reaches every model.

- **One fit per (model, table)** on all dev rows, then prediction of the test split. No model sees test rows or test labels while fitting.
- **Internal validation** (early stopping, best iteration, ensemble selection) comes from inside the dev rows, using each library's default split (pytabkit and TALENT: 20 % of the dev rows). Five-fold bagged versions of the strongest trained configurations fit on all dev rows (Amendment 5; Appendix B.3).
- **No per-table tuning** in the main comparison: library defaults (D) and the meta-tuned defaults (TD) of Holzmüller et al. (2024); foundation models at their official inference defaults and checkpoints.
- **Seeds.** Seed 0 for every cell. Seeds 1 and 2 on the fresh tables for a set of models (TabEvo, CatBoost-, LightGBM- and XGBoost-TD, TabPFN v3, TabICLv2, TabFM) estimate the seed-to-seed noise floor; these replicates never enter a ranking.
- **Features.** Numeric, binary and categorical columns; categorical columns are integer codes flagged as categorical and passed to every model that supports them natively.

Table 3: The 51 TabArena v0.1 tables (TabArena-Lite split: test = fold 0 of repeat 0 of the OpenML task; dev = the remaining rows, which every model receives in full). Grouped as used in the paper: fresh (never used in our development), held-out (consulted twice by the search, once to choose between two finalists), battery (the search's fitness tables). Task: bin = binary, mc$k$ = k classes, reg = regression. Cat. = categorical features. OpenML = task id. 51 tables.

#	Table	Task	Dev rows	Test rows	Features	Cat.	OpenML
Fresh (26)							
1	amazon-employee-access	bin	21,846	10,923	9	9	363613
2	anneal	mc5	598	300	38	32	363614
3	apsfailure	bin	50,666	25,334	170	0	363616
4	bank-customer-churn	bin	6,666	3,334	10	4	363619
5	bioresponse	bin	2,500	1,251	1,776	0	363620
6	churn	bin	3,333	1,667	19	4	363623
7	coil2000-insurance-policies	bin	6,548	3,274	85	5	363624
8	credit-g	bin	666	334	20	13	363626
9	customer-satisfaction-in-airline	bin	86,586	43,294	21	16	363628
10	diabetes130us	bin	47,678	23,840	47	39	363630
11	e-commereshippingdata	bin	7,332	3,667	10	4	363632
12	fitness-club	bin	1,000	500	6	3	363671
13	givemesomecredit	bin	100,000	50,000	10	0	363673
14	hiva-agnostic	mc3	2,563	1,282	1,617	1,617	363677
15	hr-analytics-job-change-of-data-scienti.	bin	12,772	6,386	12	10	363679
16	in-vehicle-coupon-recommendation	bin	8,456	4,228	24	22	363681
17	is-this-a-good-customer	bin	1,148	575	13	8	363682
18	jm1	bin	7,256	3,629	21	0	363712
19	kddcup09-appetency	bin	33,333	16,667	212	38	363683
20	marketing-campaign	bin	1,493	747	25	9	363684
21	mic	mc8	1,132	567	111	94	363711
22	naticusdroid	bin	4,994	2,497	86	86	363689
23	polish-companies-bankruptcy	bin	3,940	1,970	64	0	363694
24	sdss17	mc3	52,035	26,018	11	2	363699
25	seismic-bumps	bin	1,722	862	15	4	363700
26	taiwanese-bankruptcy-prediction	bin	4,546	2,273	94	0	363706
Held-out (10)							
27	another-dataset-on-used-fiat-500	reg	1,025	513	7	1	363615
28	bank-marketing	bin	30,140	15,071	13	8	363618
29	diabetes	bin	512	256	8	0	363629
30	diamonds	reg	35,960	17,980	9	3	363631
31	food-delivery-time	reg	30,300	15,151	9	3	363672
32	hazelnut-spread-contaminant-detection	bin	1,600	800	30	0	363674
33	houses	reg	13,760	6,880	8	0	363678
34	online-shoppers-intention	bin	8,220	4,110	17	7	363691
35	qsar-tid-11	reg	3,828	1,914	1,024	0	363697
36	splice	mc3	2,126	1,064	60	60	363702
Battery (15)							
37	airfoil-self-noise	reg	1,002	501	5	1	363612
38	blood-transfusion-service-center	bin	498	250	4	0	363621
39	concrete-compressive-strength	reg	686	344	8	0	363625
40	credit-card-clients-default	bin	20,000	10,000	23	3	363627
41	healthcare-insurance-expenses	reg	892	446	6	3	363675
42	heloc	bin	6,972	3,487	23	0	363676
43	maternal-health-risk	mc3	676	338	6	0	363685
44	miami-housing	reg	9,184	4,592	15	1	363686
45	physiochemical-protein	reg	30,486	15,244	9	0	363693
46	qsar-biodeg	bin	702	352	41	5	363696
47	qsar-fish-toxicity	reg	604	303	6	0	363698
48	students-dropout-and-academic-success	mc3	2,949	1,475	36	17	363704
49	superconductivity	reg	14,175	7,088	81	0	363705
50	website-phishing	mc3	902	451	9	9	363707
51	wine-quality	reg	4,331	2,166	12	1	363708

- **Missing values.** Native handling where supported; otherwise the imputation our method uses (fit-row median for numeric columns, a new code for categorical ones), recorded in the cell's configuration.
- **Input scaling.** Neural and scale-sensitive classical or kernel baselines receive numeric columns z-scored with dev-row statistics before the library's own preprocessing (Amendment 4; recorded per cell).
- **Time limit.** Two hours per fit on full data (one hour in the capped regime; the limit was not enforced on TabEvo's capped-regime cells, and one, kddcup09-appetency, took 3.2 h).
- **Compute.** GPU jobs through a slot pool on 8 NVIDIA H100 80 GB; CPU jobs with at most 8 threads on a shared 96-core host.

Table 4: The 20 high-dimensional OpenML tables of the pre-registered hypothesis H1 (Amendment 2), fixed before any of their results existed. Split: one random 80/20 dev/test split (scikit-learn `train_test_split`, `random_state=0`, stratified for classification). OpenML = dataset id. 20 tables.

#	Table	Task	Dev rows	Test rows	Features	Cat.	OpenML
High-dim. (20)							
1	ap-breast-ovary	bin	433	109	10,935	0	1165
2	ap-colon-kidney	bin	436	110	10,935	0	1137
3	arcene	bin	160	40	10,000	0	1458
4	christine	bin	4,334	1,084	1,636	37	41142
5	cnae-9	mc9	864	216	856	0	1468
6	communities-and-crime	reg	1,595	399	122	0	46286
7	dilbert	mc5	8,000	2,000	2,000	0	41163
8	gina-agnostic	bin	2,774	694	970	0	1038
9	gisette	bin	5,600	1,400	5,000	0	41026
10	internet-advertisements	bin	2,623	656	1,558	1,555	40978
11	madelon	bin	2,080	520	500	0	1485
12	mfeat-factors	mc10	1,600	400	216	0	12
13	ova-lung	bin	1,236	309	10,935	0	1130
14	qsar-tid-10980	reg	4,612	1,154	1,024	0	3277
15	qsar-tid-194	reg	4,150	1,038	1,024	0	3991
16	qsar-tid-72	reg	4,283	1,071	1,024	0	3318
17	santander-transaction-value	reg	3,567	892	4,991	0	42572
18	semeion	mc10	1,274	319	256	0	1501
19	tecator	reg	192	48	124	0	505
20	topo-2-1	reg	7,108	1,777	266	0	422

Failures. A timeout or an error is a recorded failure, never a silent skip. In the aggregates a failed cell is scored as the table's worst method (TabArena instead imputes the default random forest), and every aggregate is also reported on the tables where every compared method succeeded (complete case; Appendix I). A classification-only model on a regression table is *not applicable*: excluded, not counted as a failure. An automatic check flags any cell whose test predictions are constant.

Output. Every cell writes a result record (table, model, seed, task, number of classes, fit and query rows, status, metrics, fit and fit-plus-predict seconds, configuration, error) and its test predictions (regression) or class probabilities (classification; column k is class k). All metrics are recomputed from the saved predictions by one function.

A.4 Metrics and statistics

Metrics. Primary: accuracy (classification, argmax of the probabilities) and R^2 (regression), the objective of the search. TabArena's metrics are reported for robustness: ROC-AUC (binary; one-vs-rest macro on multiclass for reference), log-loss (multiclass) and RMSE (regression); balanced accuracy and MAE are also stored. Probabilities are used as given when they lie on the simplex; otherwise they are clipped at 10^{-4} and renormalised.

Aggregation (fixed in the evaluation plan before the results were read).

- **Primary claim family:** TabEvo against each baseline on the fresh tables, two-sided Wilcoxon signed-rank tests (Wilcoxon, 1945) with zero differences split (Demšar, 2006) and Holm correction (Holm, 1979) over the baselines, $\alpha = 0.05$; wins/ties/losses and a two-sided sign test alongside.
- **Elo:** a Bradley–Terry model on all per-table pairwise outcomes (ties count one half), random forest (D) anchored at 1000, 95 % intervals from 1,000 bootstrap resamples of the tables, with TabArena's convention for the interval (Erickson et al., 2025).
- **Normalised score** (per-table min–max over the compared methods; TabArena's median-anchored variant is also stored), **average rank** and **improvability** $(\mathrm{err} - \mathrm{err}_{\mathrm{best}})/\mathrm{err}$.
- **Friedman test** with a Nemenyi critical-difference diagram, descriptive only (Benavoli et al., 2016).
- **Parity** with the foundation models (Amendment 3): the paired mean difference (points) with a 95 % bootstrap interval over tables (10,000 resamples), and TOST equivalence tests at a margin of $\delta = 1$ point and, secondarily, at the best foundation model's seed-to-seed standard deviation. Parity is claimed only where the interval lies inside $(-\delta, \delta)$; superiority or inferiority only from the Holm-corrected Wilcoxon tests.

- **H1** (Amendment 2): on tables with many features, TabEvo's score relative to the best foundation model is higher than on the TabArena tables and grows with $\log d$. Fixed analysis: per-table difference to the best foundation model; wins/ties/losses and Wilcoxon tests against each foundation model on the high-dimensional group; the OLS slope of the difference on $\log_{10} d$ over all tables. The registered test set is the list of names above; two of its tables have fewer than 200 features (communities-and-crime, 122; tecator, 124).
- **Bagged configurations** represent their unbagged twins in the main pool (Amendment 5); the unbagged ones are compared in Appendix B.3.

A.5 Pre-registration and the dated amendments

Each rule was written down, with its date and time, before the results it governs were produced. Table 5 lists every amendment of the protocol. Other analyses were fixed in their own dated plans before their data existed: the evaluation plan, the two toy plans, the pruning plan and the held-out selection rule (Appendix D.5). Analyses decided after their results were seen are labelled *exploratory* wherever they appear.

B Baseline configurations

Every baseline is fitted by one of five runners with fixed settings and no per-table tuning; each cell records its complete configuration (estimator class, parameters, package versions, checkpoint hash, device, standardisation, imputation, large-data mechanism) in its result record. Tables 6 and 9 summarise them.

B.1 Trained baselines

Missing values. pytabkit's estimators reject NaN in continuous columns (the boosters included), so the runner imputes with the fit-row statistics our method uses (median, new category code) and records the imputation; this affects nine TabArena tables. HistGB, ExtraTrees and the foundation models receive NaN natively; TALENT applies its own policies (mean for numeric, a new category for categorical columns).

B.2 Input standardisation

The neural baselines (MLP, MLP-PLR, ResNet, FT-Transformer, TabR, RealTabR, TabM, RealMLP-TD, Ensemble-TD, the TALENT models) and the scale-sensitive classical and kernel baselines (k-NN, linear, SVM, GP, xRFM) receive every numeric, non-categorical column z-scored with the dev rows' mean and population standard deviation (NaN ignored, standard deviation floored at 10^{-12}), before the library's own preprocessing (Amendment 4). Tree ensembles and foundation models receive the raw matrix: they are invariant to monotone rescaling or bring their own preprocessing. Each cell records that it was standardised and the number of standardised columns; the analysis code rejects any cell of these models without this record.

B.3 Bagged configurations

Unbagged pytabkit models train on the 80 % of the dev rows outside their internal validation split and are not refitted, whereas TabEvo and the foundation models use all dev rows. As in TabArena, the strongest trained configurations therefore also run 5-fold bagged (Amendment 5): five models on the five cross-fitting splits of all dev rows, each early-stopped on its own held-out fold, with averaged predictions. In the main tables each of these configurations is represented by its bagged version. Table 8 compares the two.

B.4 Foundation models

TabPFN v2, v2.5 and TabICL v1.1 are earlier versions of the strongest foundation models, included so that the comparison does not hinge on one release (Amendment 1). TabICL v1.1 is classification-only; its regression cells are not applicable. The pretraining compute each foundation model's authors report is compared with the cost of our search in Appendix I.

Table 5: Dated amendments of the benchmark protocol, in the words of the rule they add; the protocol gives the full text of each. "Written before" names the results that did not yet exist when the amendment was written. Amendments 4, 5, 7, 13 and 13b change how models are fitted; the others add analyses. No amendment is numbered 14.

#	date	rule	written before
1	2026-09-25	Full data is the main regime; the 4,000-row cap becomes the secondary equal-data analysis. Exact-GP models fit at most 10,000 rows; foundation models use their official large-data mechanism; 2 h limit. TabPFN v2, v2.5 and TabICL v1.1 added.	any full-data result
2	2026-09-25	Hypothesis H1 on high-dimensional tables, its 20-table test set, split and analysis.	any high-dim. result
3	2026-09-25	Parity analysis: paired difference with bootstrap interval and TOST at $\delta = 1$ point (secondary δ: seed noise floor).	full-data and high-dim. results
4	2026-09-25	Neural and scale-sensitive classical/kernel baselines receive numeric columns z-scored with dev-row mean and standard deviation; all their cells refitted.	the reported cells of these models
5	2026-09-25	Five-fold bagged versions (`n_cv=5`) of CatBoost-, LightGBM-, XGBoost-TD, RealMLP-TD and TabM-D; they represent these configurations in the main tables.	any bagged result
6	2026-09-25	SoR variant of TabEvo: SoR/DTC predictive mean with $m = \min(n, 2000)$ inducing rows in the four GP prediction edges; reported as a separate model.	any result of it
7	2026-09-25	Full-data variant of TabEvo: the same graph on all rows (prediction pass-throughs reused; GP and metric hyperparameter searches on row subsamples; SoR/DTC with 8,000 inducing rows above 8,000 fit rows); represents TabEvo in the full-data tables. Configuration decided on the dev battery only.	any result of it
8	2026-09-25	Complementarity: equal-weight averages of saved test predictions of each foundation model with fixed partners.	any of these analyses
9	2026-09-25	Distribution-shift splits along the first principal component, with an i.i.d. control split of the same rows.	any result
10	2026-09-25	Head-weight interpretability: refits recording the stacking weight of each block; correlations with fixed table properties.	any result
11	2026-09-25	Learning curves at 10, 25 and 50 % of the dev rows (nested subsamples) on the small fresh tables.	any result
12	2026-09-25	Feature-relevance recovery on the Toy C tables from the learned GP components, compared with CatBoost and TabPFN v3 importances.	any result
13	2026-09-25	Stacking head of the full-data variant: blocks that are near-constant on the cross-fitted fitting rows are left out of the non-negative fit, and head inputs are clamped to their fitted range; rule chosen on toy data and the dev battery only, acceptance battery score ≥ 0.7891.	any benchmark cell with it
13b	2026-09-25	Probabilities of the full-data variant: $\mathrm{softmax}(s/T)$ with one temperature per fit, chosen by log-loss on the cross-fitted scores of the dev rows; the predicted class is unchanged.	any result with it
15	2026-09-26	Stack-TD: a Super Learner of CatBoost-TD, LightGBM-TD and RealMLP-TD, three-fold cross-fitted, combined by TabEvo's non-negative head; same rules and limit; run on the 36 fresh and held-out tables in both regimes. Post hoc and exploratory; it does not change any registered aggregate.	nothing (post hoc)
16	2026-09-26	Ridge-headed parent: the LLM's last lineage member (generation 3, ridge head) evaluated on the 26 fresh tables in both regimes, to test whether the authors' manual head swap changed fresh-table accuracy (Section 2.2). Post hoc, exploratory.	nothing (post hoc)
17	2026-09-26	Fresh regression group: every regression table of our data collection in no other group, split 0 of multi-split families, minus the five that duplicate a TabArena dataset (24 tables); 4,000-row regime, test split capped at 20,000 rows for every method, one seed, 1 h per cell; methods as in Table 60; primary comparison TabEvo minus Stack-TD with failures imputed, decision rule CI lower bound > 0, effect ≥ 1 point, $p < .05$. Attribution controls: GP (ARD) alone and Stack-TD with the GP as a fourth base. Registered before	nothing (post hoc)

Table 6: Trained baselines. D = library defaults, TD = the meta-tuned defaults of Holzmüller et al. (2024). Every model: seed 0, 8 CPU threads, GPU models on one H100 with a memory cap of one third of the card. Unbagged pytabkit models use pytabkit's internal validation split (20 % of the dev rows, no refit). The neural models, k-NN, the linear models, the SVM, the GP and xRFM receive standardised numeric inputs (§B.2).

method	implementation	settings	HW
Gradient-boosted trees			
XGBoost D / TD	pytabkit 1.7.3, XGBoost 2.1.4	D: 100 trees, `hist`. TD: 1,000 trees, early stopping 300 rounds (Table 7)	CPU
LightGBM D / TD	pytabkit, LightGBM 4.6.0	D: 100 trees. TD: 1,000 trees, early stopping 300 rounds	CPU
CatBoost D / TD	pytabkit, CatBoost 1.2.10	D: 1,000 trees. TD: 1,000 trees, early stopping 300 rounds	CPU
TD, bag 5	pytabkit `n_cv=5`	the TD configuration fitted on the five cross-fitting splits of all dev rows, each early-stopped on its held-out fold; predictions averaged	CPU
HistGB D	scikit-learn 1.9.0	100 iterations; early stopping on an internal 10 % split above 10,000 rows; native NaN; native categorical splits up to 255 levels	CPU
Ensemble TD	pytabkit	ensemble of the TD boosters and RealMLP-TD (standardised inputs)	GPU
Classical			
RF D	pytabkit (scikit-learn)	library defaults; fitted on all dev rows (no internal split); categorical codes ordinal	CPU
ExtraTrees D	scikit-learn	100 trees; native NaN; categorical codes ordinal	CPU
k-NN	scikit-learn	$k = 5$; numeric and binary columns standardised, categorical one-hot	CPU
Linear	scikit-learn	Ridge ($\alpha = 1$) / logistic regression ($C = 1$); inputs as k-NN	CPU
SVM	scikit-learn	RBF, $C = 1$, $\gamma =$ `scale`; Platt probabilities (internal 5-fold CV); above 20,000 rows the documented large-data route: Nyström map (1,000 components) + linear SVM, sigmoid-calibrated	CPU
GP (ARD)	scikit-learn GPR / GPC	ConstantKernel×RBF (ARD) + WhiteKernel, marginal-likelihood fit (L-BFGS-B, no restarts), initial length scale $\sqrt{d}$; at most 10,000 stratified rows (seed 0), like our exact GPs	GPU
xRFM D	pytabkit, xrfm 0.4.5	library defaults (recursive feature machines); internal 20 % split	GPU
Deep, pytabkit			
MLP D, MLP-PLR D, ResNet D, FT-Transformer D	pytabkit	the RTDL defaults (Gorishniy et al., 2021; 2022): quantile transform, AdamW, early stopping on the internal 20 % split	GPU
TabR D, RealTabR D	pytabkit	defaults of Gorishniy et al. (2024) and Holzmüller et al. (2024); exact k-NN retrieval in PyTorch	GPU
TabM D	pytabkit	$k = 32$ ensemble members, block width 512, patience 16 (Gorishniy et al., 2025)	GPU
RealMLP TD	pytabkit	meta-tuned defaults (Holzmüller et al., 2024): 3×256 units, PBLD embeddings, 256 epochs	GPU
bag 5	pytabkit `n_cv=5`	RealMLP-TD and TabM-D, five cross-fitted models	GPU
Deep, TALENT (TALENT commit `1b973ad`)			
SNN, DCNv2, NODE, TabNet, AutoInt, SAINT, ExcelFormer, ModernNCA	TALENT	TALENT's default model configs and training arguments (200 epochs, batch 1,024, best-validation checkpoint, patience 21; TabNet: its own patience 20) on an 80/20 stratified split of the dev rows	GPU

Table 7: Meta-tuned default (TD) parameters of the three boosters, as recorded in the cells' configurations: 1,000 trees with early stopping after 300 rounds on pytabkit's internal split.

	classification / regression		
parameter	XGBoost TD	LightGBM TD	CatBoost TD
learning rate	0.08 / 0.05	0.04 / 0.05	0.08 / 0.09
depth / leaves	depth 6 / 9	50 / 100 leaves	depth 7 / 9
row subsample	0.65 / 0.7	0.75 / 0.7	0.9 / 0.9 (Bernoulli)
min. child weight / data in leaf	$5 \cdot 10^{-6}$ / 2	40 / 3	–
other	colsample by level 0.9 (cls), $\lambda = 0$	colsample 1.0	ℓ_2 leaf 10^{-5}; one-hot $\leq$15 / 20 levels; random strength 0.8 / 0

Table 8: Five-fold bagged minus unbagged TD boosters, full data, primary metric in points ($\bar{\Delta}$: mean over tables; W/T/L of the bagged model). Holm-adjusted Wilcoxon $p < 0.05$ in every cell except XGBoost on fresh ($p = 0.32$); $26 + 10 + 15 + 20$ tables per configuration, no failed cell.

	fresh (26)		held-out (10)		battery (15)		TabArena (51)		high-dim. (20)	
bagged − unbagged	$\bar{\Delta}$	W/T/L	$\bar{\Delta}$	W/T/L	$\bar{\Delta}$	W/T/L	$\bar{\Delta}$	W/T/L	$\bar{\Delta}$	W/T/L
CatBoost TD	$+0.17$	17/4/5	$+0.55$	8/1/1	$+1.48$	14/0/1	$+0.63$	39/5/7	$+1.52$	17/0/3
LightGBM TD	$+0.12$	18/4/4	$+0.68$	9/0/1	$+1.90$	12/0/3	$+0.76$	39/4/8	$+1.05$	15/2/3
XGBoost TD	-0.07	14/5/7	$+0.79$	9/1/0	$+2.03$	15/0/0	$+0.71$	38/6/7	$+1.26$	17/0/3

B.5 Our method's variants

All variants of TabEvo run through one runner with seed 0, 4 CPU threads and one GPU slot, and differ only in the genome and the row budget. TabEvo as reported in the main text (the evolved genome with at most 10,000 rows; `ours_kvw_battery`) and its full-data variant (`ours_full`) are described in Appendix C; the seed (`ours_seed_kvw`) is the agent-designed program (built by an LLM agent loop, not by the evolutionary search) the search started from; the SoR variant (`ours_scalable`) is Amendment 6; speed-ups A and B (`ours_speed_a`, `ours_speed_b`) are the pruned graphs of Appendix F. Our genomes do not handle missing values, so the runner imputes as above; categorical columns are one-hot encoded inside the program.

C The evolved program in full

This section gives every node and edge of the program, the protocol that fits it to a table, and its stacking head. The evolved genome is `kvw_battery`, the final program of the search (Appendix D); the full-data variant of TabEvo ("TabEvo (full data)" in the result tables) is this genome with the fitting settings of §C.4 and the head of §C.5. The graph is drawn in Figure 2.

C.1 Genome and execution model

A genome is a JSON record of column groups, merge nodes, edges, a label map and fitting options. *Nodes* merge the blocks that reach them (concatenate, sum, elementwise product, maximum or log-sum-exp). *Edges* carry the learning: each edge has a source (a column group or an earlier node), a target node, an operator family (`lin`, `quad`, `tree`, `disc`, `gp`, `em`, `mlp`, `prod`), an output width, a role (`prediction` or `space`), an optional residual baseline, and two pieces of Python code: a predicate that decides whether the edge is active on a table, and a fitter that returns a fitted operator of the edge's family. The label map fits the head on the label node. The fitters may call any library (CatBoost, PyTorch, SciPy), but every fitted operator is exported into a native module of TabICL's synthetic-prior graph library (Qu et al., 2026), and prediction runs only these modules. An exact GP posterior mean, for example, is exported as an `em` module (a normalised radial-basis layer) whose centres are the fit rows and whose readout holds the GP weights. The genome itself never changes when a table arrives; every number the fitted predictor uses is fitted state.

Table 9: Foundation models: frozen checkpoints (hashes recorded per cell), all dev rows as context, official default inference settings, no tuning, seed 0. "Larger tables": the model's documented limits (TabPFN: read from the checkpoint's inference configuration) and the official mechanism used beyond them, with the number of full-data cells that used it; row subsampling draws a different subsample for each estimator, as the library does. All models receive categorical columns as the pandas `category` dtype where the API supports it and NaN natively.

model	package, checkpoint	inference settings	larger tables
TabICLv2 (Qu et al., 2026)	tabicl 2.1.1; `tabicl-classifier-v2-20260212`, `tabicl-regressor-v2-20260212`	8 estimators, no AMP, batch 8	no documented limit; all rows
TabICL v1.1 (Qu et al., 2025)	tabicl 2.1.1; `tabicl-classifier-v1.1-20250506`	release defaults: 32 estimators, AMP, batch 8; classification only	no documented limit; all rows
TabPFN v3 (Grinsztajn et al., 2026)	tabpfn 8.1.0; classifier `v3_default`, regressor `v3_20260417_mediumdata`	`create_default_for_version`: 8 estimators, softmax temperature 0.9; `fit_preprocessors`; no auto-scaling	limits 10^6 rows, 2,000 features: `ignore_pretraining_limits` on 4 high-dim. tables
TabPFN v2.5 (Grinsztajn et al., 2025)	tabpfn 8.1.0; `v2.5_default`	as TabPFN v3	limit 50,000 rows: row subsampling on 4 tables
TabPFN v2 (Hollmann et al., 2025)	tabpfn 8.1.0; `tabpfn-v2-classifier` (alias of `v2_default`), `tabpfn-v2-regressor`	as TabPFN v3	limits 10,000 rows, 500 features: row subsampling on 15 tables, `ignore_pretraining_limits` on 3
TabFM (Kong & Das, 2026)	tabfm 1.0.1; `google/tabfm-1.0.0-pytorch`	32 estimators, batch 1, bf16, at most 500 features used	no row limit; all rows
Mitra (Zhang et al., 2025b)	AutoGluon 1.4.0 `MitraModel`	AutoGluon defaults: fine-tuning 50 steps with early stopping on Mitra's own 80/20 split, 1 estimator	at most 10,000 rows, 500 features, 10 classes
TabDPT (Ma et al., 2025)	tabdpt 1.3.1; `tabdpt1_3`	8 ensembles, temperature 1, all dev rows as context; PCA above 128 features	no row limit
LimiX (Zhang et al., 2026)	LimiX commit `516bf39`; `LimiX-2`	repository default V2 configuration (no retrieval), as its command-line interface	at most 10 classes

The encoded table x (the built-in group `all`) holds numeric columns standardised with the fit rows' mean and standard deviation and categorical columns one-hot over the levels seen in the fit rows (centred).

C.2 Nodes and edges

The program has 10 nodes (the input x and nine merge nodes, Table 10) and 23 edges (Table 11), of which 16 are learned operators and 7 identity pass-throughs. Eight prediction blocks reach the label node Y. Relative to the seed `kernel_views_warped`, which an LLM agent loop designed before the evolutionary search, evolution added one edge, `raw_forest`; every other node, edge and the head are the seed's.

Lanes. The eight blocks into Y are three GPs on differently warped copies of the table (`kernel_Wk`, `kernel_warp`, `kernel_raw`); the gated metric GP (`read_skip`); two gated residual lanes, CatBoost and an MLP fitted to $y - K$ on H (`read_forest`, `read_network`); the exact-twin block (`twin`); and the evolved direct CatBoost lane (`raw_forest`). The three product nodes multiply a lane by a hard novelty gate that is 0 on query rows repeating a fit row,

Table 10: Nodes of the program. Every prediction block has c columns: the number of classes on classification, one on regression.

node	merge	content
x	–	encoded table (column group `all`)
W_κ (`Wk`)	concat	Kumaraswamy-warped, per-column rescaled coordinates
W_q (`W`)	concat	quantile-normal warped coordinates
M	concat	metric coordinates xL
K	concat	prediction of the GP in the metric coordinates
H	concat	$[x,\ M,$ soft novelty, hard novelty, $K]$, input of the two residual lanes
P_{forest}	product	residual CatBoost prediction $\otimes$ novelty gate
P_{network}	product	residual MLP prediction $\otimes$ novelty gate
P_{skip}	product	metric-GP prediction $K \otimes$ novelty gate
Y	concat	the eight prediction blocks; read by the stacking head

where the twin block returns the mean label of the identical fit rows instead (Appendix G examines this mechanism on duplicated rows). On tables with 500 or more columns the metric edge's marginal-likelihood search leaves L at its identity initialisation, so the metric GP then works in the standardised raw coordinates (Appendix H.5.1, design note).

C.3 Fitting protocol: staged cross-fitting

The genome's `fit_mode` is `staged`. A fit on n rows proceeds in *scopes*; the root scope holds all fit rows.

- **Space edges** are fitted in the scope and their block on the fit rows is the fitted operator's own output, so a downstream fitter works in the coordinate frame that query rows will arrive in; frames from different fits are never mixed.
- **Prediction edges** are cross-fitted: the scope's rows are split into F folds (`KFold`, shuffled with the table seed; $F = 3$). For each fold, a child scope on the other rows fits the edge's *entire upstream prefix* (encoding, space edges, other prediction edges) and predicts the held rows. The block on the fit rows is assembled from these held-out predictions, so no downstream fitter sees a row's prediction from a model that saw its label. The operator used at query time is the scope's own fit on all its rows.
- **Residual edges** (`forest`, `network`) are fitted to the targets minus the cross-fitted block of their baseline (`kernel_copy`) in the same scope.
- **Targets.** Regression blocks and residuals are in original target units; classification blocks are scores on uncentred one-hot targets.
- **Head.** The label map is fitted once, in the root scope, on the cross-fitted blocks that reach Y (§C.5).
- **Seeding.** Each edge's fit is seeded from the table seed, the scope size and the edge id, and the folds from the table seed, so removing an edge does not change the random state of any other edge.

Because child scopes cross-fit their own prediction edges, the scopes nest: the program's deepest chain ($K \to H \to$ residual lanes $\to Y$) gives $1+3+9+27 = 40$ scopes per fit. With the framework option `reuse_prediction_passthroughs`, an identity edge that copies an already cross-fitted prediction (`kernel_copy`, `skip`) reuses that block instead of refitting its prefix one level deeper, which leaves 13 scopes and the predictions bit-identical on all 15 battery tables (step s01 of the pruning loop, Table 24). The full-data variant uses this option.

C.4 Fitting settings of the benchmarked method

The full-data variant differs from the evolved genome only in how it is fitted (Table 12; Amendments 7, 13 and 13b, Table 5). Every setting was chosen on the dev battery (and toy data for the head), never on a benchmark table.

Inducing-point solve. Above $m = 8{,}000$ fit rows (in any scope), each of the four GP prediction edges replaces the exact solve by the subset-of-regressors/DTC predictive mean (Quiñonero-

Table 11: The 23 edges of the program in genome order (the fitter code of each edge is in its `transform_code` field). All predicates are `return True`. *GP kernel* of edges 2, 3, 5, 7: powered exponential $k(x,x') = \sigma_f^2 \exp\big(-(\|x-x'\|/\ell)^q\big)$ with one lengthscale ℓ, $q = 0.5 + 1.5\,\mathrm{sigmoid}(\cdot) \in (0.5, 2)$, noise $\sigma_n^2 \geq 10^{-4}$ and jitter 10^{-6}; standardised targets; hyperparameters by Adam on the exact marginal likelihood in float64 on a row subsample, then on a larger one (sizes in Table 12); the posterior mean uses every fit row. Edge 1 feeds per-column lengthscales into edge 2 (an ARD kernel through the warp); edges 6–7 are a GP in a learned linear metric. *Role*: a `prediction` block is cross-fitted on the fit rows (§C.3), a `space` block is the in-scope fit's own output.

	edge	source → target	family, role	fitted operator
1	`front_Wk`	$x \to W_\kappa$	mlp, space	learned input warping (Snoek et al., 2014) with the Kumaraswamy CDF (Kumaraswamy, 1980) $1-(1-u^{a_j})^{b_j}$ on each min–max-scaled column, times a per-column inverse lengthscale; fitted by exact GP marginal likelihood (float64, Adam): lengthscales, signal, noise and kernel exponent with the warp at identity, then the warp shapes under a log-normal prior (s.d. 0.5, $\lvert\log a_j\rvert, \lvert\log b_j\rvert \leq 1.2$)
2	`kernel_Wk`	$W_\kappa \to Y$	em, prediction	exact GP posterior mean (kernel below)
3	`kernel_raw`	$x \to Y$	em, prediction	exact GP posterior mean on the encoded table
4	`warp`	$x \to W_q$	mlp, space	per-column quantile-normal warp: piecewise-linear through 32 empirical quantile knots to their normal scores, flat outside (an exact two-layer ReLU net)
5	`kernel_warp`	$W_q \to Y$	em, prediction	exact GP posterior mean
6	`metric`	$x \to M$	lin, space	full linear metric L ($d \times d$, initialised at I) with an exponential kernel, signal and noise, by exact marginal likelihood (float64, Adam)
7	`kernel`	$M \to K$	em, prediction	exact GP posterior mean in the metric coordinates
8	`raw`	$x \to H$	lin, space	identity
9	`coordinates`	$M \to H$	lin, space	identity
10	`soft_novelty`	$M \to H$	em, prediction	$1/(1+\sum_i e^{4-d_i/s})$ over the fit rows, s a quarter of the median nearest-neighbour distance: $\approx$0.02 on a repeated row, $\approx$1 far from the data
11	`novelty`	$x \to H$	em, prediction	hard novelty: 0 on a row that repeats a fit row, 1 otherwise (kernel width 10^{-6} of the feature scale)
12	`kernel_copy`	$K \to H$	lin, prediction	identity (the cross-fitted metric-GP prediction)
13	`forest`	$H \to P_{\text{forest}}$	tree, prediction	CatBoost (Prokhorenkova et al., 2018) fitted to $y - K$ (residual of `kernel_copy`); one RMSE forest per target column; depth $\mathrm{round}(\log_2(n/8)) \in [4, 12]$, $n/15$ trees in $[200, 2000]$; exported exactly into the native oblivious-tree module
14	`network`	$H \to P_{\text{network}}$	mlp, prediction	MLP fitted to $y - K$: three hidden layers of 256 ReLU units on standardised H, AdamW (lr 10^{-3}, weight decay 10^{-4}), batch 256, at most 200 epochs, early stopping on a held tenth of the fit rows (patience 24)
15	`skip`	$K \to P_{\text{skip}}$	lin, prediction	identity
16	`gate_forest`	$x \to P_{\text{forest}}$	em, prediction	hard novelty gate (as edge 11)
17	`read_forest`	$P_{\text{forest}} \to Y$	lin, space	identity
18	`gate_network`	$x \to P_{\text{network}}$	em, prediction	hard novelty gate
19	`read_network`	$P_{\text{network}} \to Y$	lin, space	identity
20	`gate_skip`	$x \to P_{\text{skip}}$	em, prediction	hard novelty gate
21	`read_skip`	$P_{\text{skip}} \to Y$	lin, space	identity
22	`twin`	$x \to Y$	em, prediction	exact-twin block: the mean target of the fit rows identical to the query row, 0 on any other row
23	`raw_forest`	$x \to Y$	tree, prediction	**evolved**: CatBoost on the encoded table, learning rate 0.05, depth as edge 13; the number of trees is chosen on a deterministic 15 % holdout (growth in chunks of 50 up to 2,000, patience 8 chunks; classification by held-out accuracy with a 10^{-7} cross-entropy tie-break, regression by squared error), then the model is refitted on all fit rows with that number

Table 12: The evolved program as selected and as benchmarked on full data. Search sizes are upper bounds on a GPU (row subsamples drawn with the table seed). Battery fitness is the mean 3-fold score over the 15 battery tables (accuracy or R^2): 0.78929 as stored by the evolution run, 0.78960 re-scored in the pruning harness, 0.78997 for the full-data variant with its head (the acceptance threshold fixed in Amendment 13 was 0.7891).

	TabEvo as evolved	TabEvo (full data)
rows fitted	stratified subsample, $\leq$ 10,000	all dev rows
GP solve (edges 2, 3, 5, 7)	exact on all fit rows	exact up to 8,000 fit rows; SoR/DTC with 8,000 inducing rows above
GP hyperparameter search	$\leq$ 4,096 rows $\times$ 150 steps, then $\leq$ 8,192 $\times$ 40	$\leq$ 1,024 rows $\times$ 75 steps, then $\leq$ 2,048 $\times$ 20
metric L search	$\leq$ 6,000 rows $\times$ 300 steps, then $\leq$ 20,500 $\times$ 60	$\leq$ 1,024 rows $\times$ 150 steps, then $\leq$ 2,048 $\times$ 40
Kumaraswamy warp search	$\leq$ 4,096 $\times$ 150, then $\leq$ 8,192 $\times$ 100	unchanged
prediction pass-throughs	refitted (40 scopes)	reused (13 scopes)
stacking head	non-negative block stacking	+ exclusion of near-constant blocks, input clamping, temperature softmax (§C.5)
battery fitness (3-fold CV)	0.78929 (0.78960 re-scored)	0.78997

Candela & Rasmussen, 2005; Titsias, 2009): $\mu(x) = k_m(x)^\top w$ with $w = (sK_{mm} + K_{mn}K_{nm})^{-1}K_{mn}t$, $s = \sigma_n^2 + 10^{-6}$ (the exact solve's diagonal), t the standardised targets, and m inducing rows drawn uniformly without replacement with the table seed. It is computed in float64 in whitened form with K_{mn} accumulated over chunks of 8,192 rows, so no $n \times n$ matrix is formed. The hyperparameters are fitted exactly as in the exact path, and with $n \leq m$ the code path is the exact one character for character (unit tests: bit-identical outputs; with every row inducing, the SoR weights equal the exact posterior weights). On heloc (4,000 rows) with $m = 2{,}000$, the SoR mean of `kernel_raw` tracks the exact mean with RMSE 0.030 of its standard deviation and correlation 0.9995. The scalable variant of Amendment 6 is the same construction with $m = 2{,}000$ on the unchanged genome (battery fitness 0.78801).

C.5 The stacking head

Let $B_1, \ldots, B_8$ be the cross-fitted blocks in Y on the n fit rows (each $n \times c$), T the canonical targets (one-hot on classification, y on regression), and tildes denote column-centring over the fit rows. The head fits one weight per block, shared by the block's columns, and a per-column intercept:

$$\min_{w \geq 0} \Big\| \sum_{j \in \mathcal{K}} w_j \tilde{B}_j - \tilde{T} \Big\|_F^2, \qquad \mathcal{K} = \big\{\, j : \ \mathrm{RMS}(\tilde{B}_j) \ \geq \ \tau \, \mathrm{RMS}(\tilde{T}) \,\big\}, \quad \tau = 10^{-3}, \tag{2}$$

solved by non-negative least squares on the stacked block columns; $w_j = 0$ for $j \notin \mathcal{K}$ and the intercept is $\bar{T} - \sum_j w_j \bar{B}_j$. Non-negativity lets the head drop a lane (weight 0) rather than cancel two lanes against each other. Three properties complete the head of the full-data variant (Amendments 13 and 13b):

1. **Only informative blocks are weighted.** A block whose centred cross-fitted columns have an RMS below τ times the RMS of the centred targets carries no usable variation on the fitting rows, and its non-negative weight is not identified; it receives weight 0. The threshold is scale-free (relative to the target). Example: when no dev row repeats another, the twin block is (near-)constant on the cross-fitted rows.
2. **Inputs stay in their fitted range.** Every head input column is clamped to its $[\min, \max]$ over the cross-fitted fitting rows before the weights are applied, so no block is read outside the range on which its weight was fitted. On the fitting rows the clamp is the identity.
3. **Calibrated probabilities.** On classification the head outputs $p = \mathrm{softmax}(z/T)$ with z its class scores and one temperature $T = e^a$ per fit, $a \in [-9, 7]$ minimising the cross-entropy of the cross-fitted scores on the fitting rows (bounded scalar search). T acts on each row's score differences monotonically, so the predicted class, and hence accuracy, is unchanged.

The head is exported as a two-layer native module: the first layer maps each input column onto $[0, 1]$ over its fitted range and clamps it; the second holds the rescaled block weights, the intercept and, on classification, the tempered softmax.

How the rule was chosen. The rule was chosen on the dev battery and on the Toy A tables only, and written down before any benchmark table was fitted with it. On the battery (15 tables $\times$ 3 folds) every informative block has an RMS ratio of at least 0.0465, while blocks without variation on the fitting rows (the twin block on Toy A) have ratios below 10^{-7}; any τ in $[10^{-6}, 10^{-2}]$ gives identical results on every battery table, and $\tau = 10^{-3}$ lies between the two with margins of 10^4 and 46. With this head, Toy A (a one-dimensional function with a jump, $n = 300$) is fitted with grid R^2 of 0.934, 0.931, 0.954, 0.944 and 0.970 on data seeds 0–4, and the largest absolute error at query points within 10^{-5} of a training point is 0.22–0.59; on the battery the fitness is 0.78997 (Table 12). TabEvo as evolved and the other pre-registered variants keep the evolved head (the non-negative stacking of Equation 2 over all blocks, with probabilities from the metric's clip-and-renormalise rule).

C.6 What the fitted program costs

A profile of the evolved program on four battery tables (Appendix F.2) attributes most of the fitting time to the GP-type hyperparameter searches (the metric and the four GP edges; the exact solve and the export take less than 1 s per table) and to the CatBoost lanes, while the MLP lane, the gates, the twin block, the identity copies and the head together take a few per cent. The settings of Table 12 reduce the geometric-mean fitting time on the battery from 244 s to 107 s per table (steps s01–s03 of the pruning loop, Table 24; one H100 per table, 3 outer folds). Per-table times on the benchmark are in Appendix I.

D The evolution run

TabEvo is the result of one search run of 11.7 h wall-clock time. Every number in this section is computed read-only from the stored records of that run. Fitness is the mean 3-fold cross-validated score over the 15 battery tables (accuracy or R^2).

D.1 Set-up

The diff language. The mutation operator never writes a whole genome. It writes the complete child as lists of `keep` entries (a parent element by identifier, with optional replacements of individual fields such as one edge's code) and `new` entries (full records); omitted elements are deleted. The JSON schema of the answer is generated from the parent, so the identifiers that can be kept are enumerated. Before a child is fitted, the framework resolves identifiers, validates the topology (unique identifiers, existing endpoints, forward edges, residual baselines), compiles every piece of code, and checks the declared structural intent (Table 14).

Course of the run. The run had six attempts separated by five supervised restarts (6.6 min of downtime in total), at which the operators adjusted the task text, the parent selection or the scheduler (Table 15). Each restart cancelled the evaluations in flight, which were rerun; cancelled evaluations account for 14.6 of the 54.5 child-evaluation hours.

D.2 Statistics and trajectory

Almost every proposal became a valid program: 5 of 59 did not parse and 1 of 51 evaluated children failed at fit time, three of these six on a single corrupted character. The best-so-far moved three times, each time through a direct CatBoost lane: to 0.79070 at 2.0 h (an added lane), 0.79172 at 3.2 h (the lane `raw_forest`) and 0.79254 at 7.2 h (`raw_forest` stopped on held-out accuracy). 43 of the 50 valid children beat the seed `kernel_views_warped`; 29 improved on their own parent, 5 were flat and 16 lower (median change $+0.68 \cdot 10^{-3}$). The 54 LLM children came from 21 distinct parents, and 12 of the 50 valid children fall into 4 groups of identical fitness (functionally identical programs). Only 5 of the 54 proposals cited a memory card.

Table 13: Search configuration of the run. Seed fitness values are those measured in the run.

component	setting
fitness	mean over the 15 battery tables of the 3-fold CV score of the program on the table's dev rows (accuracy / R^2); the five tables above 4,000 dev rows as fixed 4,000-row subsamples; each table fitted in its own worker on a shared pool of 16 slots on 8 H100; fixed seeds (deterministic)
seed programs	`kernel_views_warped` (stacking head, 0.78804), `kvg_kwarp` (ridge head, 0.78918), `kvg_kwarp_nn_f5` (stacking head, 0.78708), all built by the agent loop
mutation	one parent per child; the LLM returns a structured diff (below), with its justification
LLM roles	insight analyst, lineage analyst and mutation operator: `gpt-5.6-sol`, reasoning effort high; memory cards: written by `gpt-5.6-luna` from evaluated children (bootstrapped from the evaluated seeds) and offered to the mutation operator when it edits other lineages
feedback in prompts	per battery table: score, fold s.d., balanced accuracy or log-loss, head weight of every prediction block, fitting behaviour; the child's per-table change against its parent; validator errors with the failing line of code; the genome rendered as graph semantics. No held-out or fresh table ever entered a prompt
archive	MAP-Elites (Mouret & Clune, 2015), one island of up to 1,200 programs; cells = (mean regression score, mean classification score) on the battery, 60×50 linear bins on $[0.70, 0.82] \times [0.74, 0.84]$ (about 0.002 wide around the seed), upper bounds growing on demand
parent selection	softmax over each elite's fitness gap to the best, temperature 0.0025; a parent's weight $\times 0.2$ for every child of it still queued or in evaluation (or drawn in the last 900 s); from attempt 5 only programs with the non-negative stacking head
concurrency	up to 3 children in evaluation, 4 concurrent LLM calls

Table 14: Structural intents of the diff and their enforcement.

declared intent	check applied to the child, beyond the common topology and compile checks
`add_group`	more column groups than the base parent
`add_node`	more nodes than the base parent
`add_edge`	more edges than the base parent
`simplify`	fewer groups, nodes and edges in total
`local_edit`, `reroute`	none (descriptive)

Table 15: The six attempts of the run, in order: duration in hours, and children persisted and evaluated during the attempt.

attempt	hours	change at the start of the attempt	children	evaluated
0	1.29	initial launch: three seeds, tables fitted in parallel, memory cards	8	1
1	1.20	task text: multiclass CatBoost reference; one sentence removed	8	8
2	2.65	parent selection penalises parents with children in flight	15	14
3	2.38	task text: measured per-table headroom; in-flight factor 0.2	15	15
4	2.33	scheduler slots kept free for parent refreshes; task text: facts on CatBoost lanes	8	10
5	1.75	parents restricted to the stacking head; stacking-head variants of the best programs added by the operators	3	3

Table 16: Statistics of the run. GPU-hours are estimated from a pre-run measurement of 1.7 worker-hours per child with 2 workers per GPU; the bound integrates the occupied slots of the 16-slot pool. Both LLMs were accessed through a subscription CLI, so token counts are exact and no monetary cost is derivable. About 5 min of mutator time per child against 42 min of evaluation: the run was bounded by fitting.

Run	
wall-clock, 6 attempts	11.7 h
LLM proposals	59
code did not parse (3 syntax errors, 2 stray characters)	5
children: LLM-proposed + added by the operators	54 + 3
children evaluated / valid	51 / 50
not scored (4 discarded in attempt 5, 2 in evaluation at the stop)	6
evaluated children per hour	4.4
distinct parents	21
evaluation time per child, median (range)	42 min (6–102)
Compute	
GPU-hours used, estimate (upper bound)	≈46 (≤86)
GPU-hours reserved (8 GPUs × wall-clock)	94
share of wall-clock with at least one child in evaluation	92 %
LLM (mutator stack / memory)	
calls	217 / 190
mutator stack: insight / mutation / lineage roles	141 / 59 / 17
prompt tokens (cached)	9.91 M (1.48 M) / 7.86 M (0.75 M)
completion tokens (incl. reasoning)	754 k / 61 k
mean latency per call	78 s / 19 s
total LLM time	4.7 h / 1.0 h
failed calls, transient retries	0, 0
Battery fitness	
seed `kernel_views_warped`	0.78804
best stacking-head child (finalist A)	0.79068
best child overall (ridge head)	0.79254
final program `kvw_battery`	0.78929

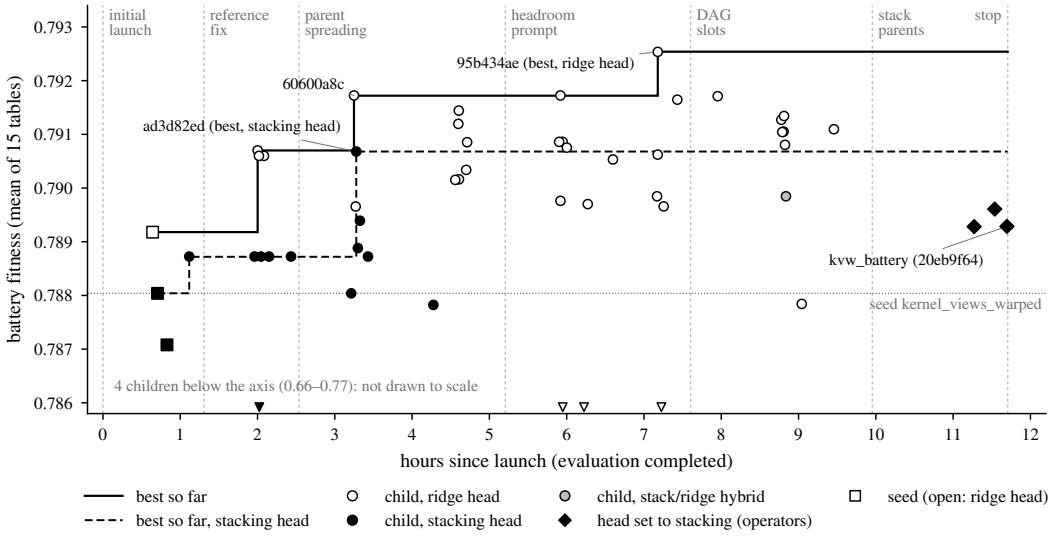


Figure 5: Battery fitness of every evaluated valid child against the time its evaluation completed, with the best-so-far over all programs (solid) and over stacking-head programs (dashed); dotted verticals mark the attempts of Table 15. Four children with fitness 0.66–0.77 are drawn at the lower edge; 50 valid children and 3 seeds in total.

Table 17: Mutation classes, from rules on the genome diff of each child against its parent (a child can belong to several classes; indented rows split a class by lane family or head direction). Δ is the child's battery fitness minus its parent's; scored = evaluated and valid; flat: $|\Delta| \leq 10^{-4}$. The column $\Delta \leq -10$ counts children at least 0.01 below their parent, which the mean* excludes. New best: children that raised the run's best-so-far. In total 57 children, 50 scored.

			Δ fitness (10^{-3})					
class	children	scored	median	mean*	best	up/flat/down	$\Delta \leq -10$	new best
new lane	20	17	$+0.74$	$+0.96$	$+2.55$	14/1/2	–	2
CatBoost lane	17	14	$+0.79$	$+1.06$	$+2.55$	13/1/0	–	2
MLP lane	3	3	-0.26	$+0.47$	$+1.96$	1/0/2	–	–
lane removal	3	3	$+0.70$	$+1.33$	$+1.96$	2/0/1	1	–
fitter change	17	14	$+0.06$	$+0.10$	$+0.81$	5/4/5	1	1
CatBoost lane	12	9	$+0.09$	$+0.17$	$+0.81$	4/3/2	–	1
MLP lane	5	5	-0.18	-0.04	$+0.70$	1/1/3	1	–
head change	14	13	$+0.76$	-0.09	$+1.97$	7/0/6	1	–
ridge to stacking (LLM)	3	3	-1.87	-1.87	-1.87	0/0/3	1	–
ridge to stacking (oper.)	3	3	-2.06	-2.45	-2.04	0/0/3	–	–
stacking to hybrid (LLM)	1	1	$+0.96$	$+0.96$	$+0.96$	1/0/0	–	–
stacking to ridge (LLM)	7	6	$+1.56$	$+1.51$	$+1.97$	6/0/0	–	–
gate change	2	2	-129.12	–	-125.64	0/0/2	2	–
representation change	4	4	$+1.26$	$+0.73$	$+1.94$	3/0/1	–	–
input warp	2	2	0.00	0.00	$+1.56$	1/0/1	–	–
lane input rerouted	2	2	$+1.45$	$+1.45$	$+1.94$	2/0/0	–	–
all children	57	50	$+0.68$	$+0.42$	$+2.55$	29/5/16	4	3

Table 18: Ancestry of `kvw_battery`. The final genome equals the seed `kernel_views_warped` (fitness 0.78804) plus the `raw_forest` lane: $+0.00125$ on the battery.

gen.	program	head	change	fitness	Δ
1	`b13a7dbb`	ridge	seed `kvg_kwarp`: the lanes of `kernel_views_warped` with an affine ridge head	0.78918	–
2	`60600a8c`	ridge	new lane `raw_forest`: CatBoost on the raw table, number of trees chosen on a holdout	0.79172	$+0.00255$
3	`95b434ae`	ridge	`raw_forest` stops on held-out accuracy (not cross-entropy) on classification	0.79254	$+0.00081$
4	`20eb9f64`	stacking	head set to the seed's non-negative stacking head (by the operators)	0.78929	-0.00325

Adding a direct CatBoost lane was the most reliable move of the run: 13 of the 14 scored additions improved on their parent and none was lower (Table 17).

D.3 Lineage of the final program

The final program `kvw_battery` is three steps from its root (Table 18, Figure 6). Its genome is exactly the agent-designed seed `kernel_views_warped` plus one edge, the direct CatBoost lane `raw_forest` with a holdout-selected number of trees and accuracy-based stopping on classification: same nodes, same other 22 edges, same head. The last step, which set the head of the program to the seed's non-negative stacking head, was added by the operators when parents were restricted to the stacking head (attempt 5), and was evaluated like every other child.

D.4 Two mutations in the mutator's words

The mutation operator returns structured output whose fields `justification` and `changes[].description` state its rationale. Each program below is linked to its LLM call by exact equality of the justification with one logged response; every quotation is an exact substring of that response, and the released call logs give the line of each.

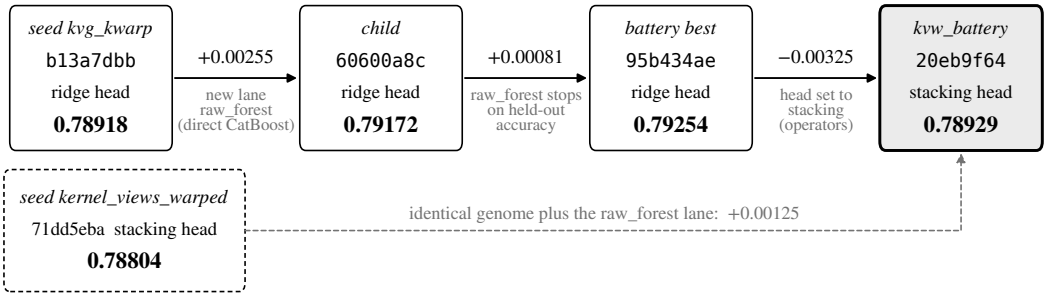


Figure 6: Lineage of the final program with the battery fitness of every step.


```
{"archetype": "Proven Pattern Extension", "justification": "...",
 "insights_used": [...], "insight_ids_used": [...], "base_parent": "A",
 "card_ids_used": [], "changes": [{"description": "...", "explanation": "..."}],
 "fit_code": null, "fit_mode": null, "structural_intent": "add_edge",
 "groups": [], "nodes": [{"kind": "keep", "id": "Wk", "edits": null}, ··· 8 more],
 "edges": [{"kind": "keep", "id": "front_Wk", "edits": null},
   {"kind": "keep", "id": "kernel_Wk", "edits": null},
   ··· 20 more kept edges: kernel_raw, warp, kernel_warp, ..., read_skip, twin
   {"kind": "new", "id": "raw_forest", "source": "group:all", "target": "Y",
    "family": "tree", "residual_edge": null, "width": "classes",
    "carries": "prediction", "predict_code": "return True",
    "transform_code": "118 lines, decoded; 1–20: imports, 15 % held-out split, settings
21      stopper = EarlyStop(patience=8, min_delta=1e-5)
22      total = 0
23      while total < max_trees:
24          candidate = make_model(chunk)
25          if model is None:
26              candidate.fit(X_train, target_train)
27          else:
28              candidate.fit(X_train, target_train, init_model=model)
29          model = candidate
30          total += chunk
31          pred = model.predict(X_held, prediction_type='RawFormulaVal')
    lines 32–42: loss on the held-out rows (cross-entropy, or squared error in regression)
43          if stopper.observe(total, loss):
44              break
45      return int(max(chunk, stopper.best_step))
    lines 46–118: refit the selected number of trees on all rows; export them as a native tree
   ", "rationale": "Independent original-target CatBoost lane with..."}],
 "label": {"kind": "keep", "edits": null}}
```


justification
"The verified `tried: direct-target forest lane` cluster improved all three attempts (mean delta +0.00146), so the p73 middle-band exception supports exploiting it further."

changes[0].description
"Added raw_forest with fixed row-count→validation-selected duration: tree-competitive concrete and Miami contain nonlinear target signal unavailable after kernel residualization."

measured, 15 development tables
fitness 0.78918 → 0.79172 (+0.00255), a new best; 12 tables up, 2 down, 1 unchanged

concrete*	+0.0117 R^2
miami-housing*	+0.0049 R^2
qsar-fish-toxicity	+0.0051 R^2
qsar-biodeg	+0.0043 acc.

* named in the description

Figure 7: One mutation from the search, in full: the structured diff that added the edge `raw_forest` (a CatBoost model on the raw row whose number of trees is chosen on a holdout and then refitted on all rows) to its parent, the LLM's own justification quoted verbatim, and the measured effect on the development tables. This edge is the last block of the program in Figure 2.

(a) The direct CatBoost lane. At 0.92 h, in the first batch of the run, a child of the seed added an edge named `raw_forest`: a CatBoost lane from the raw table straight to the label, next to the kernel-residual forest.

> "Added raw_forest with kernel-residual→original-target training: concrete and airfoil tree references beat kNN by 0.1803 and 0.1543, indicating partition-specific signal."
> "A separate raw-target tree lane can represent discontinuities independently, while the existing non-negative stacker can suppress it on datasets where that hypothesis fails."

It raised fitness from 0.78804 to 0.78872 (+0.00069). At 2.31 h the mutator added the lane again, to the ridge-head seed `kvg_kwarp`, now choosing the number of boosting rounds on held-out rows and refitting on all rows (generation 2 of Table 18), and cited the earlier children of this kind:

> "The verified 'tried: direct-target forest lane' cluster improved all three attempts (mean delta +0.00146), so the p73 middle-band exception supports exploiting it further."
> "Added raw_forest with fixed row-count→validation-selected duration: tree-competitive concrete and Miami contain nonlinear target signal unavailable after kernel residualization."

Fitness rose from 0.78918 to 0.79172 (+0.00255, the largest single step of the run; 12 tables up, 2 down), with the largest gains on the tables the mutator named: concrete +0.0117 and miami-housing +0.0049 in R^2, qsar-fish-toxicity +0.0051 in R^2 and qsar-biodeg +0.0043 in accuracy. This lane, with accuracy-based stopping added in the next step, is the one the final program adds to its seed.

The stacking head uses the lane where the argument predicts: on the battery its mean weight is 0.27 on the regression tables and 0.06 on the classification tables, and it is correlated with the table's gain from evolution (Pearson $r = 0.71$, $p = 0.003$; Spearman $\rho = 0.38$, $p = 0.16$; 15 tables; Appendix H.5.2).

(b) A memory card carries an idea across lineages. At 4.76 h the memory author (`gpt-5.6-luna`) wrote a card from a stacking-head child (later finalist A, §D.5) that had replaced the gated residual network by a direct per-feature additive network:

> "…replace a high-capacity joint residual MLP with a low-width additive MLP that gives each feature independent hidden units, selecting training duration on an internal holdout and refitting on all rows, because removing cross-feature parameter coupling reduces estimation variance while retaining univariate nonlinearities."

At 7.22 h the card was offered, in a randomised probe, to the mutator editing a program of another lineage (ridge head, 0.79034). The mutator cited the card and made two changes:

> "Swapped 'network' joint 256-unit layers→per-feature 8-unit residual subnetworks: six-to-81-feature tables expose variance from unrestricted cross-feature estimation."
> "Dropped direct-target 'additive_network': duplicating additive fits across only 33–267 validation rows makes their correlated head estimates unnecessarily unstable."

The child improved from 0.79034 to 0.79104 ($+0.00070$; 9 tables up, 5 down).

D.5 Selection between the finalists on the held-out tables

Two stacking-head programs were finalists: A (the best stacking-head child: a direct CatBoost lane and a direct per-feature additive network added, the gated residual MLP lane removed) and B (the lanes of the best ridge-head child with the stacking head, generation 4 of Table 18). The selection rule was written down before any held-out number was read: a finalist that fails a table or loses more than 0.02 to the seed on one is dropped; of the rest, the one with the higher held-out mean wins, provided that mean is at least the seed's; ties within 0.001 go to the higher battery fitness. The ten held-out tables were scored with the battery protocol (3-fold cross-validation on their dev rows, at most 4,000 rows). Both finalists beat the seed on every held-out table they did not tie (Table 19); B has the higher held-out mean ($+0.00756$ over the seed against $+0.00553$ for A) and became `kvw_battery`. Much of B's margin comes from food-delivery-time ($+0.0540$); on the other nine tables the mean gain is $+0.0024$ for B and $+0.0023$ for A.

D.6 Scope of the evidence

- **One run, one evaluation per program.** The evaluation is deterministic (functionally identical children reproduce the same fitness to within 10^{-8}), but there are no seed replicates, and fitness differences of about 10^{-3} are small next to the mean per-table fold standard deviation (about 0.019). The held-out selection and the benchmark are the evidence that the result transfers. There is no variance over repeated search runs.
- **Operator decisions.** The restarts changed the task text, the selection and the scheduler (Table 15), and the final head step was added by the operators; the run is not a single stationary process.
- **Compute accounting.** Per-table fit times were not recorded during the run, so GPU-hours are estimated (Table 16). The development of the seed programs is not included in the cost of the search.

D.7 The agent loop from scratch

The starting genome has one node `Y`, one edge `kernel_raw` (the exact GP of the evolved program, code copied verbatim) and the NNLS block-stacking head. Each candidate makes one edit and is evaluated on the 15 development tables under the unchanged protocol; it is accepted if its score exceeds the best accepted score by more than 0.0005. The loop evaluated two alternative single edits of the start in its first round (Table 20).

Table 19: Per-table scores of the seed and changes of the two finalists, battery protocol (3-fold CV on the dev rows; †: dev rows capped at 4,000). B is the final program `kvw_battery`. On the battery B gains on 8 regression tables (+0.0026 mean R^2) and loses 0.0003 mean accuracy on 7 classification tables.

battery (fitness tables) table		seed	A−seed	B−seed
airfoil	R^2	.9506	+.0001	.0000
concrete	R^2	.9111	+.0087	+.0119
insurance	R^2	.8496	−.0010	+.0001
miami†	R^2	.9037	+.0037	+.0042
protein†	R^2	.5815	−.0027	+.0010
qsar-fish	R^2	.6288	+.0080	+.0031
supercond.†	R^2	.8920	−.0010	+.0003
wine-quality	R^2	.4879	+.0003	−.0001
blood-transf.	acc	.7590	+.0080	−.0020
credit-default†	acc	.8115	+.0032	+.0015
heloc†	acc	.7140	+.0023	+.0008
maternal-health	acc	.8003	+.0015	+.0015
qsar-biodeg	acc	.8675	+.0057	−.0043
students-dropout	acc	.7728	−.0007	−.0003
website-phishing	acc	.8902	+.0033	+.0011
mean		.78804	+.00264	+.00125
up / down			11 / 4	10 / 4

held-out (selection) table		seed	A−seed	B−seed
diamonds†	R^2	.9703	+.0011	+.0013
food-delivery†	R^2	.2426	+.0349	+.0540
houses†	R^2	.8270	+.0005	+.0015
qsar-tid-11	R^2	.7244	+.0021	+.0105
used-fiat-500	R^2	.8431	+.0060	+.0009
bank-marketing†	acc	.8900	+.0005	+.0008
diabetes	acc	.7520	+.0059	+.0039
hazelnut	acc	.9363	+.0013	.0000
online-shoppers†	acc	.9060	+.0008	+.0008
splice	acc	.9483	+.0024	+.0019
mean		.80398	+.00553	+.00756
up / down			10 / 0	9 / 0

Table 20: **From one Gaussian process to the evolved program** on the 15 development tables: mean score, its regression and classification parts, and the paired difference to the single GP with a 95 % bootstrap interval, tables up/down and the Wilcoxon p; it01 and it02 are the two single edits of it00 evaluated in the agent loop's first round.

	Battery score			vs. it00 (15 tables)		
Genome	all	reg. (8)	cls. (7)	Δ [95% CI]	up/down	p
it00: one exact GP + NNLS head	0.7674	0.7386	0.8003	–	–	–
it01: it00 + direct CatBoost lane	0.7823	0.7637	0.8036	+1.50 [0.56, 2.66]	13/2	.002
it02: it00 + Kumaraswamy-warp GP lane	0.7856	0.7671	0.8068	+1.82 [0.77, 3.12]	11/2	.004
evolved program (kvw_battery)	0.7896	0.7780	0.8029	+2.22 [0.84, 3.82]	11/4	.004

E Ablations

The ablations remove one part of the evolved program at a time, or replace its head, and refit the result on the same tables with the same fitting rules (pre-registered in the evaluation plan, Appendix A.5). They measure what each prediction block, the gates and the stacking head contribute on tables the search never saw.

E.1 The ablated programs

A script derives every ablated genome from `kvw_battery` and checks that it parses and compiles and that every surviving node and edge is byte-identical to the original and in the original order. A block is removed by deleting its edge(s) into Y and then every node that no longer has a path to Y, together with the edges into such nodes; nothing else changes (code, widths, residual links, fitting options, head). Since every edge's fit is seeded from its own identifier and the cross-fitting folds from the table seed, a removal does not change the random state of any surviving edge: a variant differs from the program only by the removed part and by what the head does without it.

Reproducibility check. The variant without `raw_forest` is the seed genome up to two rationale strings, so it must reproduce the seed's benchmark cells exactly. It does: on all 12 tables where both were compared, the maximum absolute difference of the primary metric is 0.

Table 21: The 13 ablated programs. Node counts exclude the input.

variant	change against `kvw_battery` (9 merge nodes, 23 edges)	nodes / edges
leave one prediction block out (the head stacks the remaining seven)		
w/o `kernel_Wk`	remove `kernel_Wk`, `front_Wk` and node W_κ	8 / 21
w/o `kernel_warp`	remove `kernel_warp`, `warp` and node W_q	8 / 21
w/o `kernel_raw`	remove `kernel_raw`	9 / 22
w/o `read_skip`	remove the gated metric-GP block (`read_skip`, `skip`, `gate_skip`, P_{skip})	8 / 20
w/o `read_forest`	remove the residual CatBoost lane (`read_forest`, `forest`, `gate_forest`, P_{forest})	8 / 20
w/o `read_network`	remove the residual MLP lane (`read_network`, `network`, `gate_network`, P_{network})	8 / 20
w/o `twin`	remove the exact-twin block	9 / 22
w/o `raw_forest`	remove the evolved direct CatBoost lane; this is the seed genome	9 / 22
head and gates		
ridge head	the head of the best ridge-head child (Table 18, gen. 3): one affine ridge map on all block columns, signed weights, no weight sharing within a block	9 / 23
mean head	fixed weight $1/m$ per block (m blocks present), same intercept rule	9 / 23
w/o gates	remove `gate_forest`, `gate_network`, `gate_skip`: every lane is read on every row, including rows that repeat a fit row	9 / 20
single-family references (unchanged head)		
CatBoost only	only `raw_forest` into Y	1 / 1
GP only	only the four GP blocks (`kernel_Wk`, `kernel_warp`, `kernel_raw`, `read_skip` without its gate)	6 / 9

E.2 Protocol

Every variant runs through the benchmark's own runner, seed 0, on the 26 fresh and 10 held-out tables with the dev rows capped at 4,000 (the 25 tables with at most 4,000 dev rows are identical to the full data). The reference is TabEvo as evolved on the same data, so each comparison differs only in the program. A failed cell is scored as in the benchmark. A leave-one-block-out variant whose test predictions equal the reference's (maximum absolute difference below 10^{-6}) is one on which the head had given the removed block weight 0.

E.3 Results

Table 22 reports every variant against the evolved program.

Two further pieces of evidence isolate single parts under controlled conditions. On duplicated rows (Toy B, Appendix G), the variant without gates is markedly less accurate on the repeated rows than the program, which shows what the gates are for. In the pruning loop (Appendix F), removing the metric-GP block or the Kumaraswamy GP block lowered the battery score below the tolerance (0.78809 and 0.78655 against the floor of 0.7891), whereas the two residual lanes could be removed together at an unchanged score, so the several GP views are the part of the program that carries its accuracy on the development tables. How the head distributes its weight over the blocks, table by table, is analysed in Appendix H.5.2.

Table 22: Ablations of the evolved program (TabEvo as evolved, the reference), primary metric in points: $\bar{\Delta}$ is the variant minus the reference (negative: the removed part helped), W/T/L the variant's wins, ties and losses, p the two-sided Wilcoxon test, Holm-corrected over the variants. Regime: 4k cap (the data the ablations were fitted on). Tables per row: the sum of W/T/L; 128 variant cells on fresh + held-out in total.

	Fresh			Held-out			Fresh + held-out		
variant − evolved program	$\bar{\Delta}$	W/T/L	p	$\bar{\Delta}$	W/T/L	p	$\bar{\Delta}$	W/T/L	p
seed (= w/o direct CatBoost lane)	−0.07	5/12/8	1.000	−0.63	3/2/5	1.000	−0.23	8/14/13	1.000
w/o GP on warp-likelihood warp	−0.09	2/3/3	1.000	−0.10	0/2/2	1.000	−0.09	2/5/5	1.000
w/o GP on quantile warp	0.14	2/6/0	1.000	0.30	1/2/1	1.000	0.19	3/8/1	1.000
w/o GP on raw features	0.13	3/5/0	1.000	0.09	1/2/1	1.000	0.12	4/7/1	1.000
w/o metric-GP lane	0.06	2/4/2	1.000	−0.06	2/1/1	1.000	0.02	4/5/3	1.000
w/o residual CatBoost lane	0.02	3/4/1	1.000	0.01	2/1/1	1.000	0.01	5/5/2	1.000
w/o residual MLP lane	−0.03	0/6/1	1.000	−0.00	0/1/1	1.000	−0.03	0/7/2	1.000
w/o twin-mean lane	–	–	–	–	–	–	–	–	–
w/o direct CatBoost lane	−0.03	1/5/2	1.000	0.17	2/1/1	1.000	0.03	3/6/3	1.000
w/o novelty gates	–	–	–	–	–	–	–	–	–
mean head (no stacking)	–	–	–	–	–	–	–	–	–
ridge head (no stacking)	−2.85	2/3/3	1.000	−0.26	1/1/2	1.000	−1.98	3/4/5	1.000
GP lanes only	–	–	–	–	–	–	–	–	–
direct CatBoost lane only	–	–	–	–	–	–	–	–	–

F PRUNING FOR SPEED

After the evolved program was selected, we made it cheaper to fit with a hand-run profile–change–re-score loop. The loop is engineering, not a test of a hypothesis: its written brief fixed the procedure, the acceptance rule and the final candidates before it started, and every decision used the 15 development-battery tables only. No fresh, held-out or high-dimensional result was read before the loop ended. The benchmark uses the loop in two ways (§F.5): its first three accepted steps are part of the full-data variant of TabEvo (Amendment 7), and its final genomes, `speed_a` and `speed_b`, are reported as the secondary rows TabEvo speed-up A and B.

F.1 PROTOCOL

- **Scoring.** The unchanged fitness evaluation of the search: 3 outer cross-validation folds per battery table, seed 0, the five tables above 4,000 rows capped at 4,000. Score = mean per-table fitness (accuracy or R^2), exactly the search's battery fitness.
- **Timing.** One process per table and one table per H100 (at most 8 tables at once, 4 CPU threads, `PYTHONHASHSEED=0`, CUDA context warmed before the clock). Time = geometric mean over the 15 tables of the summed fit-plus-predict seconds of the 3 folds.
- **Reference.** `kvw_battery` re-scored in this harness: 0.78960 (the search stored 0.78929; per-table differences up to ± 0.0014 come from thread count and scheduling), $T_{\mathrm{ref}} = 244.3\,\mathrm{s}$.
- **Changes.** One named, exact text edit of the genome per step; every candidate is parsed, compiled, deterministic (seed 0) and keeps the non-negative stacking head. Every experiment is logged, kept or not.
- **Acceptance.** Steps s01–s08 used the brief's rule: score $\geq$ current best $- 0.0005$, score $\geq 0.78929 - 0.001$, and time $\leq 0.9\times$ the current accepted genome. Steps s09–s18 used a fixed score anchor instead: score ≥ 0.7891 (= re-scored reference $- 0.0005$) and $\geq 0.78929 - 0.001$, with the same time bar. The two earlier rejections that pass the fixed anchor (s05, s07) were re-tested on the current accepted genome (s10–s12).
- **Stop and final candidates** (fixed in the brief). Stop after 6 consecutive rejections, which happened with s13–s18, about 2 h into the loop. `speed_a` = the last accepted genome; `speed_b` = the fastest logged genome with score $\geq 0.78929 - 0.003$.

Table 23: Where the reference genome's fit time goes: share (%) of the fit-plus-predict time summed over the 3 outer folds, by component, on four battery tables (fit rows per fold × features; total seconds). "Three GP views" sums the GPs on the Kumaraswamy warp, the quantile warp and the raw row. Components below 1 % on every table (novelty gates, twin, identity copies, projections, cross-fitted predictions, the NNLS head) are omitted. One profiled run per table.

component	supercond. 2667×81, reg. 758 s	students-dropout 1966×36, 3 cl. 517 s	heloc 2667×23, bin. 399 s	concrete 458×8, reg. 141 s
direct CatBoost (`raw_forest`)	54.1	18.1	9.4	11.0
residual CatBoost (`forest`)	8.8	32.3	7.8	1.8
linear metric (`metric`)	15.5	20.7	35.9	42.4
GP in the metric (`kernel`)	9.6	13.0	22.3	26.3
three GP views	6.3	7.5	14.9	8.8
Kumaraswamy warp (`front_Wk`)	3.1	3.9	7.1	6.9
residual MLP (`network`)	0.9	0.7	0.8	1.1
time in the 27 deepest scopes	13	18	30	46

F.2 Where the time goes

Table 23 profiles the reference genome on four battery tables of different size and task, with lap timers in the fitter bodies (numerics unchanged). Three facts set the order of the changes.

- **Nested refits of identity copies.** Staged fitting cross-fits every prediction edge (3 inner folds) and refits its whole upstream prefix in each child scope (§C). The metric-GP prediction K reaches H and P_{skip} through identity copies, which were themselves cross-fitted, so the metric and its GP were refitted one level deeper than any reader needs: $1+3+9+27 = 40$ fitting scopes per outer fold. The deepest 27 took 13 % (superconductivity) to 46 % (concrete) of a table's time. The framework option `reuse_prediction_passthroughs` reuses the cross-fitted K block instead (13 scopes) and leaves every prediction bit-identical.
- **GP fitters are hyperparameter search.** In the metric and the four GP views the Adam marginal-likelihood search is more than 97 % of the edge's time; the exact solve and the export take under 1 s per table. On small scopes a step is latency-bound (≈1.4 ms at 135–458 rows), so the number of steps is the lever there.
- **CatBoost cost** scales with the number and depth of trees: the direct lane's holdout selection of the tree count and its refit on all rows are 58 % and 42 % of the lane on superconductivity.

F.3 The accepted chain and all logged genomes

Five changes were kept on top of the prediction-preserving s01 (Table 24): two remove work (the nested refits, and both residual lanes on H with everything only they use) and three buy cheaper budgets (GP and metric searches on row subsamples with fewer steps, cheaper CatBoost lanes, two instead of three inner folds). The final genome `speed_a` fits 6.5× faster than the reference at battery score 0.78941. Table 25 lists all 19 logged genomes and Figure 8 plots score against time.

What stays. After s12 every remaining single change either lowered the score below the bar or saved less than 10 %. The GP views the head uses carry accuracy: without the metric-GP lane the battery score falls by 0.0013 from s12 (s17), without the Kumaraswamy GP lane by 0.0029 (s18). Removing the raw-row GP view (s13, score 0.78999) keeps the score but saves only 3.5 %.

`speed_b`. The fastest logged genome within 0.003 of the stored fitness is s18 (= s12 without the Kumaraswamy GP lane): 26.2 s per table, 9.3× faster than the reference, battery score 0.78655, 5 merge nodes and 10 edges.

F.4 The full and the pruned graph

Table 26 compares the two genomes and Figure 9 draws the pruned one in the style of the main-text Figure 2. The speed-up is paid on regression: `speed_a` is 0.0028 below the re-scored reference

Table 24: Accepted chain of the pruning loop (15 battery tables, 3 outer folds each). Score is the battery fitness; reg./cls. are its means over the 8 regression and 7 classification tables; time is the geometric mean over tables of fit-plus-predict seconds summed over the folds; the last two columns are time ratios to the parent and to the reference. s01 leaves every per-table score bit-identical. Exploratory (engineering loop under a brief fixed before it started).

step	change	score	reg.	cls.	time (s)	×par.	×ref.
s00	reference: `kvw_battery` unchanged	0.78960	0.77799	0.80288	244.3	1.000	1.000
s01	reuse cross-fitted passthroughs (40 → 13 scopes)	0.78960	0.77799	0.80288	154.9	0.634	0.634
s02	metric search on 1024 rows × 150 steps, then 2048 × 40 (was all rows, 300 + 60)	0.79088	0.77833	0.80523	136.1	0.879	0.557
s03	GP searches (4 views) on 1024 × 75, then 2048 × 20 (was all rows, 150 + 40)	0.79050	0.77786	0.80494	106.8	0.785	0.437
s06	CatBoost: direct lane learning rate 0.1 (was 0.05); residual forest 1 tree per 30 rows in [100, 1000] (was 15, [200, 2000])	0.79041	0.77776	0.80487	92.5	0.866	0.378
s09	CatBoost depth ≤ 6 in both lanes (was $\mathrm{round}(\log_2(n/8)) \leq 12$)	0.79058	0.77809	0.80486	76.9	0.831	0.315
s10	remove both residual lanes on H (CatBoost, MLP, their gates; H pruned): 6 merge nodes, 12 edges	0.79068	0.77643	0.80698	51.5	0.670	0.211
s12	s10 with 2 inner cross-fitting folds (was 3) = `speed_a`	0.78941	0.77516	0.80569	37.5	0.728	0.153

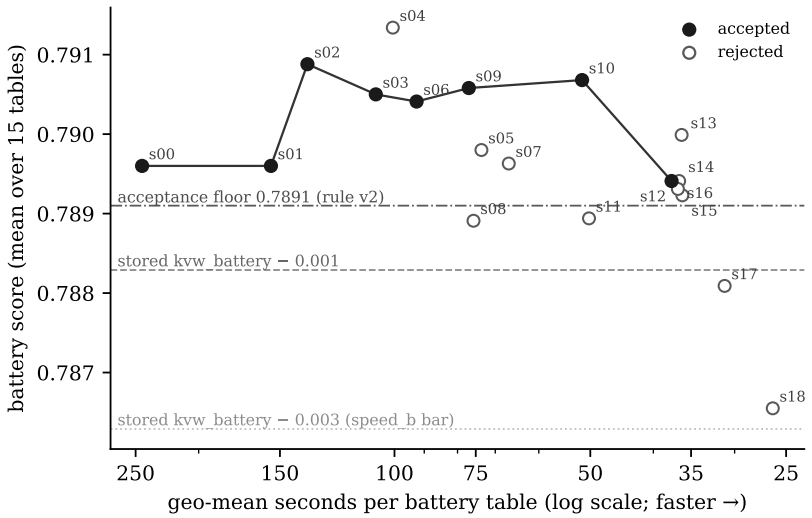


Figure 8: Battery score against geometric-mean fit-plus-predict time of every genome of the pruning loop (Table 25).

on the 8 battery regression tables (physiochemical-protein -0.019) and 0.0028 above it on the 7 classification tables.

F.5 How the benchmark uses the loop

- **TabEvo (full data).** Amendment 7 builds the full-data entry of TabEvo on the genome after s03: the reuse of cross-fitted passthroughs (s01) and the row-subsampled metric and GP searches (s02, s03). The graph, the CatBoost lanes and the 3 inner folds are those of `kvw_battery`; the large-n solve and the head are described in §C.
- **TabEvo speed-up A and B** (`speed_a`, `speed_b`). Both genomes were fitted on the 26 fresh tables with the benchmark runner, exactly as TabEvo as evolved (the 11 fresh tables with at most 4,000 dev rows on all rows; the other 15 on the full data with its 10,000-row

Table 25: All 19 genomes of the pruning loop, in the order they were logged. Kept: y = accepted. Otherwise the binding criterion is given: "time" = time ratio to the parent above 0.9; "score" = below the score bar in force (s00–s08: current best $-\,0.0005$, marked v1; s09–s18: the fixed anchor 0.7891). One evaluation per genome.

id	parent	change	score	time (s)	×par.	kept
s00	–	reference	0.78960	244.3	1.000	y
s01	s00	reuse cross-fitted passthroughs	0.78960	154.9	0.634	y
s02	s01	metric search on row subsamples	0.79088	136.1	0.879	y
s03	s02	GP searches on row subsamples	0.79050	106.8	0.785	y
s04	s03	direct CatBoost learning rate 0.1	0.79134	100.6	0.942	n, time
s05	s03	remove both residual lanes on H	0.78980	73.5	0.688	n, score (v1)
s07	s03	2 inner folds	0.78963	66.7	0.625	n, score (v1)
s06	s03	cheaper CatBoost (rate 0.1, half the residual trees)	0.79041	92.5	0.866	y
s08	s06	halve the metric and Kumaraswamy searches	0.78891	75.6	0.817	n, score (v1)
s09	s06	CatBoost depth ≤ 6	0.79058	76.9	0.831	y
s11	s09	2 inner folds	0.78894	50.2	0.654	n, score
s10	s09	remove both residual lanes on H	0.79068	51.5	0.670	y
s12	s10	2 inner folds (with s10)	0.78941	37.5	0.728	y (= `speed_a`)
s13	s12	remove the raw-row GP view `kernel_raw`	0.78999	36.2	0.965	n, time
s14	s12	`cholesky_ex` in the Adam loops (bit-identical scores)	0.78941	36.5	0.974	n, time
s15	s12	Kumaraswamy search on 1024, then 2048 rows	0.78923	36.1	0.963	n, time
s16	s12	one CatBoost fit replays the tree-count selection	0.78931	36.7	0.978	n, time
s17	s12	remove the metric-GP lane (`read_skip`; M, K pruned)	0.78809	31.1	0.829	n, score
s18	s12	remove the Kumaraswamy GP lane (`kernel_Wk`)	0.78655	26.2	0.699	n (= `speed_b`)

Table 26: Structure, fitting settings, battery score and fit time of the full evolved graph (`kvw_battery`) and of the pruned one (`speed_a`). Node counts include the input x. Time is fit plus prediction summed over the 3 outer folds, geometric mean over the 15 battery tables.

	full (`kvw_battery`)	pruned (`speed_a`)
nodes / edges	10 / 23	7 / 12
learned edge operators	16	10
prediction blocks into the head	8	6
residual lanes on H (CatBoost, MLP)	2	0
search rows, 4 GPs and metric	all rows[a]	1024, then 2048
Adam steps, 4 GPs / metric	150+40 / 300+60	75+20 / 150+40
direct CatBoost depth, learning rate	$\mathrm{round}(\log_2(n/8)) \leq 12$, 0.05	$\mathrm{round}(\log_2(n/8)) \leq 6$, 0.1
inner cross-fitting folds	3	2
fitting scopes per fit	40	3
battery score	0.7896 (stored 0.7893)	0.7894
regression / classification	0.7780 / 0.8029	0.7752 / 0.8057
fit time per table	244.3 s	37.5 s (6.5× faster)

[a] Up to 4096 then 8192 rows (GPs) and 6000 then 20,500 rows (metric) on a GPU; every battery fold has fewer fit rows.

cap), and are reported as secondary rows. Each has results on 23 of the 26 fresh tables; their median fit-plus-predict times there are 49 s and 34 s on the shared GPUs of the benchmark node (Table 62). The median of 646 s of TabEvo as evolved is over a different set (46 TabArena tables), so the ratio is not a speed-up measurement.

Caveats. Every genome was evaluated once. Re-scoring the unchanged reference moved the battery mean by $+0.0003$, and per-table moves of ± 0.003 to ± 0.009 between near-identical genomes

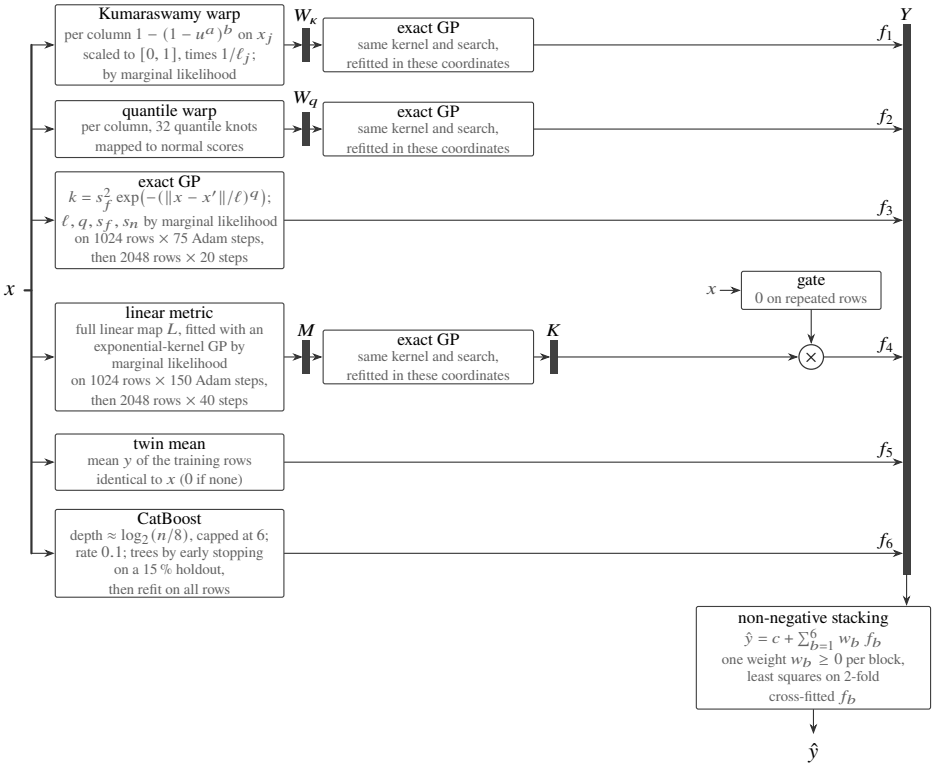


Figure 9: The pruned graph `speed_a`, with every node and edge, drawn as Figure 2. Both residual lanes are gone: the node H with its five inputs, the CatBoost and the MLP fitted to $y - K$, their gates and products. Four exact GPs remain, on the Kumaraswamy-warped row, the quantile-warped row, the raw row and the metric coordinates $M = Lx$; the gate zeroes the metric GP's prediction K on query rows that repeat a training row, where the twin block returns the mean label of the identical training rows. The GP and metric hyperparameters are searched on 1024 and then 2048 random training rows; each GP posterior still uses every training row. The direct CatBoost has depth at most 6 and learning rate 0.1, its tree count chosen on a holdout (accuracy in classification, squared error in regression). The head is fitted on blocks cross-fitted over 2 inner folds.

are common, so battery differences of about 0.001 are within evaluation noise. The battery tables are capped at 4,000 rows and every fit sees two thirds of them, so the timings describe small tables.

G Controlled toy experiments

Four synthetic studies isolate properties of TabEvo that the benchmark tables mix together: the fit of one smooth function with a jump (Toy A), repeated rows (Toy B), irrelevant features (Toy C) and the number of rows (Toy D). Two written plans, one for Toys A and B and one for Toys C and D, fixed the designs, models and analyses before any toy data existed. Additions made after results were seen are marked exploratory.

Common rules. Every cell runs through the benchmark's own, unmodified runners with the benchmark's schema (§A): fit on all development rows (train ∪ validation), predict the test split, model seed 0, default settings, the Amendment-4 standardisation for the networks and kernel baselines. In Toys B and C the "seeds" are independent data sets, so standard deviations measure data-sampling spread. Toys A and B: 172 of 172 cells succeeded; Toys C and D: 260 of 260 (one CUDA out-of-memory error of TabPFN v3 on a shared GPU was retried once, as the plan allows). Unless stated otherwise "ours" is TabEvo, the evolved program as selected.

G.1 Toy A: one smooth function with a jump

Design (pre-registered). $f(x) = \sin(2\pi x)$ for $x < 0.5$ and $\sin(2\pi x) + 1.5$ for $x \geq 0.5$; 300 development rows with $x \sim U[0,1]$ and $y = f(x) + \mathcal{N}(0, 0.1^2)$; the test set is a 1,000-point grid

Table 27: Toy A: test R^2 against the noise-free f on a 1,000-point grid, per data set (300 development rows each; bold: best in the column). TabEvo is its full-data variant (Amendments 7 and 13; 13b changes only classification probabilities), fitted on all rows; the last row is its largest absolute error over the queries $x_j + \delta$, $\delta \in \{0, \pm 10^{-7}, \pm 10^{-6}, \pm 10^{-5}\}$, at all 300 training inputs x_j. Baselines: the benchmark runners. Data set 0 pre-registered; data sets 1–4 and the evaluation of the full-data variant exploratory. 20 runner cells and 5 diagnostic fits.

data set	0	1	2	3	4
TabEvo (full data)	0.934	0.931	0.954	0.944	0.970
CatBoost-TD	0.941	**0.956**	0.955	0.963	0.978
TabPFN v3	0.965	0.948	**0.980**	**0.983**	**0.990**
TabICLv2	**0.972**	0.951	0.970	0.972	0.985
exact GP (ARD)	0.938	0.922	0.934	0.931	0.944
TabEvo: max. |error| near training inputs	0.22	0.21	0.48	0.56	0.59

on $[0, 1]$ scored against the noise-free f, so the score measures recovery of f. Models: TabEvo, CatBoost-TD, TabPFN v3, TabICLv2 and the exact ARD GP. The plan fixed data seed 0; data seeds 1–4 (exploratory) use the identical generator.

Result. Table 27 gives the grid R^2 of the full-data variant of TabEvo, whose stacking head leaves blocks that are near-constant on the cross-fitted rows out of the non-negative fit and clamps every head input to its fitted range (§C.5), next to the baselines. On this one-dimensional smooth function TabEvo is at the level of the exact ARD GP and CatBoost-TD (above the GP on 4 of 5 data sets, below CatBoost-TD on all 5), and below both foundation models. Its median absolute error on the grid is 0.033 ± 0.005 over the five data sets, against 0.036 ± 0.004 (CatBoost-TD), 0.016 ± 0.004 (TabPFN v3), 0.020 ± 0.003 (TabICLv2) and 0.029 ± 0.004 (ARD GP). Its largest grid errors, 0.70 to 1.52 on data sets 0–3, lie at the jump of f, where every model errs; on data set 4 the largest is 0.49. At the 300 training inputs themselves and at queries within 10^{-5} of them (the region where the exact-duplicate lanes act), its largest absolute error is 0.21–0.59 per data set.

G.2 Toy B: repeated rows

Design (pre-registered). Binary classification with 8 numeric features (scikit-learn's generator with 6 informative and 2 redundant features, 2 clusters per class and 20 % of labels redrawn at random, so about 10 % of labels are wrong and a repeated row's label cannot be recovered from x by a smooth model alone); 2,000 development rows and 1,000 fresh test rows per data set, 3 data sets. *Variant B:* a fraction $p \in \{0, 0.1, 0.25, 0.5\}$ of the test rows is replaced by exact copies (x and label) of development rows, nested in p; the development rows contain no duplicates. *Variant B2* (secondary, also pre-registered): the development rows contain the same fraction of duplicates as the test rows. Models: ours, its ablations without the twin lane and without the three novelty gates (§E), CatBoost-TD, TabPFN v3, TabICLv2 and k-NN, on all 21 tables (147 cells).

Mechanism. Three parts of TabEvo act on repeated rows. The *novelty gates* multiply the metric-GP block and the two residual lanes on H by 0 on a query row that repeats a training row; the residual lanes are fitted on out-of-fold values, so at a training point the gates leave the prediction to the ungated lanes: the three GP views, which nearly interpolate the training label, and the direct CatBoost lane. The *twin lane* returns the mean label of a query row's identical training rows. The *stacking head* weights the twin lane only when the development rows themselves contain repeats, because only then is its cross-fitted block informative.

Results. Tables 28 and 29, Figures 10 and 11.

- **Ranking quality.** Ours has the highest ROC-AUC of all models at every $p > 0$ in variant B: 0.903, 0.912 and 0.936 at $p = 0.1$, 0.25 and 0.5, against the next best 0.901 (TabICLv2), 0.900 (TabICLv2) and 0.924 (CatBoost-TD).
- **Accuracy.** Ours is 0.014 below TabPFN v3 at $p = 0$, within 0.01 of the foundation models at $p = 0.1$ and 0.25, and 0.01 above them at $p = 0.5$. CatBoost-TD, which memorises, gains most on the repeated rows (0.93–0.95 there).

Table 28: Toy B, variant B (repeated rows only in the test split): accuracy (mean and standard deviation over 3 data sets) at test duplication fraction p, and mean ROC-AUC. Ours without the twin lane is identical to ours on all 12 tables. Bold: best in the column. Pre-registered; 84 of the 147 Toy B cells.

	accuracy				ROC-AUC	
model	$p = 0$	$p = 0.1$	$p = 0.25$	$p = 0.5$	$p = 0$	$p = 0.5$
ours	$0.863_{\pm.011}$	$0.867_{\pm.006}$	$0.866_{\pm.007}$	$0.885_{\pm.018}$	0.896	**0.936**
ours without gates	$0.863_{\pm.011}$	$0.837_{\pm.026}$	$0.796_{\pm.054}$	$0.737_{\pm.126}$	0.896	0.783
CatBoost-TD	$0.842_{\pm.019}$	$0.850_{\pm.020}$	$0.862_{\pm.018}$	$\mathbf{0.891}_{\pm.010}$	0.883	0.924
TabPFN v3	$\mathbf{0.877}_{\pm.016}$	$\mathbf{0.876}_{\pm.014}$	$\mathbf{0.868}_{\pm.019}$	$0.877_{\pm.013}$	0.896	0.911
TabICLv2	$0.871_{\pm.013}$	$0.871_{\pm.010}$	$0.865_{\pm.016}$	$0.875_{\pm.010}$	**0.900**	0.914
k-NN	$0.844_{\pm.019}$	$0.846_{\pm.020}$	$0.842_{\pm.022}$	$0.852_{\pm.016}$	0.876	0.903

Table 29: Toy B: accuracy on the repeated test rows in variant B (left), and variant B2, where the development rows contain the same fraction of repeats (right; mean accuracy over all test rows, ROC-AUC at $p = 0.5$, accuracy on the repeated rows). Standard deviations over the 3 data sets are 0.009–0.031 in B2. Bold: best in the column. Pre-registered; 147 cells.

	B: repeated rows			B2: repeats in development and test				
				accuracy			AUC	repeated
model	0.1	0.25	0.5	0.1	0.25	0.5	0.5	rows
ours	0.913	0.899	0.917	0.872	0.888	0.922	0.966	**1.000**
ours without twin	0.913	0.899	0.917	0.867	0.884	0.921	0.965	**1.000**
ours without gates	0.613	0.617	0.621	0.873	0.887	0.923	0.965	**1.000**
CatBoost-TD	**0.933**	**0.939**	**0.949**	0.847	0.865	0.905	0.945	0.97–0.99
TabPFN v3	0.880	0.861	0.883	**0.887**	**0.898**	0.920	0.969	**1.000**
TabICLv2	0.883	0.869	0.887	0.876	0.891	**0.925**	**0.970**	**1.000**
k-NN	0.867	0.851	0.866	0.836	0.831	0.840	0.905	0.86–0.88

- **What the gates contribute.** On the repeated rows ours reaches 0.90–0.92 accuracy; the same program without the gates reaches 0.61–0.62, and its overall accuracy falls from 0.863 at $p = 0$ to 0.737 at $p = 0.5$. On fresh rows the two are identical, as they must be.
- **The twin lane.** In variant B the head gives the twin lane weight 0 (seed-0 refits, every p) and ours equals the ablation without it on all 12 tables (maximum absolute probability difference 0). In variant B2 the twin lane receives weights 0.50, 0.32 and 0.47 at $p = 0.1$, 0.25 and 0.5 (seed 0); every model except CatBoost-TD and k-NN, with or without the twin lane, then classifies the repeated rows perfectly, and the twin lane adds $+0.001$ to $+0.005$ accuracy, within the spread over data sets.
- **Pre-registered predictions.** The twin-weight and equality prediction for variant B and the positive twin weight in variant B2 held. The plan expected ours to do worse than the no-gates ablation on repeated rows of variant B, and expected the jump on repeated rows in B2 to be specific to the twin lane; neither held (see above).

G.3 Toy C: irrelevant features

Design (pre-registered). Friedman's function #1, $10\sin(\pi u_1 u_2)+20(u_3-0.5)^2+10u_4+5u_5$ with $u_j = \Phi(x_j)$, on 5 signal columns, plus $d-5$ irrelevant columns; every column is marginally $\mathcal{N}(0,1)$, so no model can tell signal from noise by its distribution, and the column order is randomly permuted per table. $d \in \{5, 20, 100, 500, 2000\}$, 500 development and 2,000 test rows, 3 data sets, nested in d (within a data set only the added irrelevant columns change). Classification: $y = 1[f > \text{median}]$ with 5 % label flips (Bayes accuracy 0.95); regression: $y = f + \mathcal{N}(0,1)$ (mean test-oracle R^2 0.961). Models: ours, TabPFN v3, TabICLv2, TabFM, CatBoost-TD, RealMLP-TD and the exact ARD GP (210 cells). The foundation models run with the main benchmark's settings: TabPFN v3's pretraining feature limit, read from its checkpoint, is 2,000, so $d = 2000$ is within it; the TabFM runner's fixed limit of 500 features caps TabFM at 500 of the 2,000 columns; TabICLv2 documents no limit (pretrained on tables with at most about 100 features).

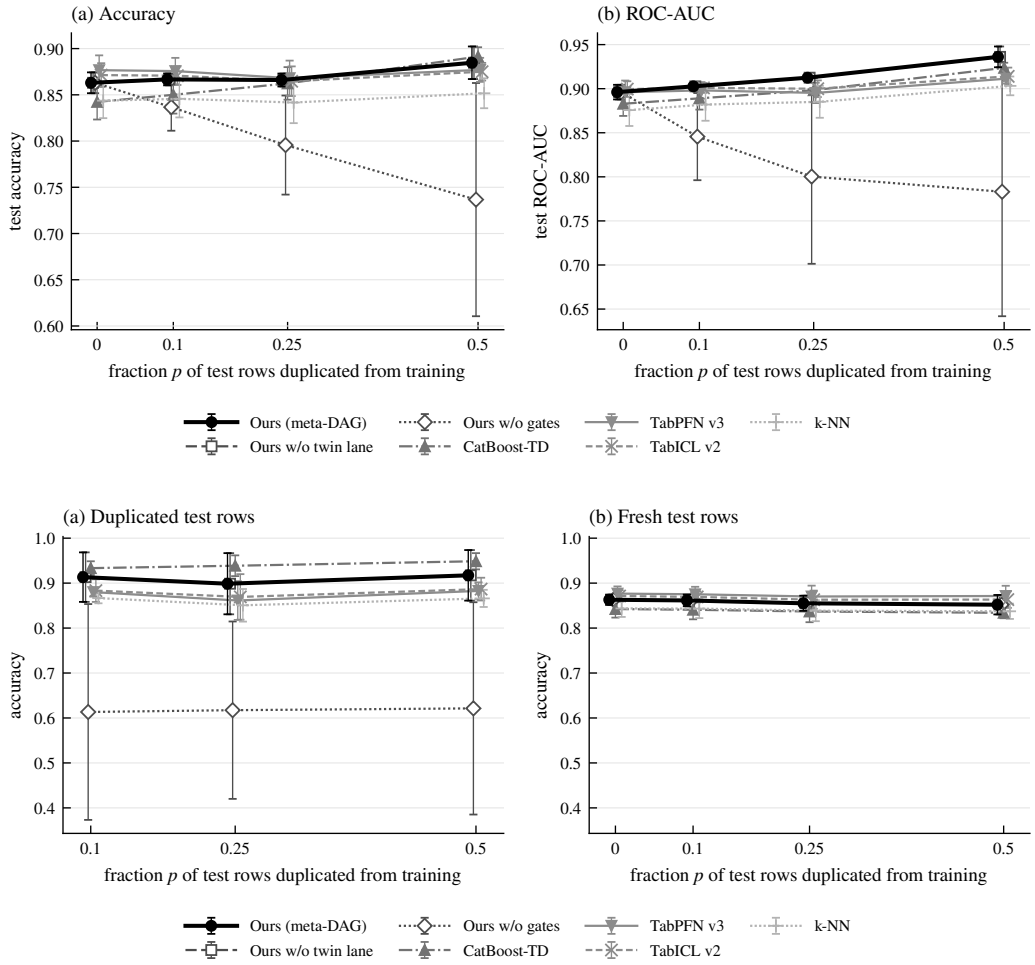


Figure 10: Toy B, variant B: accuracy and ROC-AUC against the fraction p of test rows repeated from the development rows (top), and accuracy on the repeated and on the fresh test rows (bottom); mean and standard deviation over 3 data sets.

Pre-registered toy analogue of H1. $\mathrm{gap}(d, s)$ = ours minus the best foundation model on table (d, s), in points. Primary metric accuracy. "Supported" if the OLS slope of the gap on $\log_{10} d$ is positive for each of the 3 data sets and the mean gap at $d = 2000$ exceeds the one at $d = 5$; "contradicted" if both go the other way for every data set; otherwise "mixed". AUC and regression R^2 are secondary.

Results (Tables 30–32, Figures 12 and 13).

- **Robustness to irrelevant features.** From $d = 5$ to $d = 2000$ ours loses 8.3 accuracy and 8.6 R^2 points. At $d = 2000$ it is the best model in the study on every metric: accuracy 0.800 (6.8 points above the best foundation model's mean, 1.7 above CatBoost-TD), ROC-AUC 0.881, R^2 0.862 against 0.669 for the next best (CatBoost-TD) and 0.33–0.39 for the foundation models.
- **Not simply a GP.** The exact ARD GP loses 13.8 accuracy and 28.6 R^2 points over the same range. CatBoost-TD is the most robust baseline in classification (-6.3 points) but starts 3.8 points below ours at $d = 5$.
- **The shape is a threshold.** Up to $d = 500$ the foundation models tolerate irrelevant columns better than ours (the gap moves from -2.1 to -5.5 accuracy points and from -0.4 to -5.9 R^2 points; TabPFN v3 and TabFM lose at most 0.2 R^2 points up to $d = 500$). The sign changes between $d = 500$ and $d = 2000$, where all three foundation models degrade

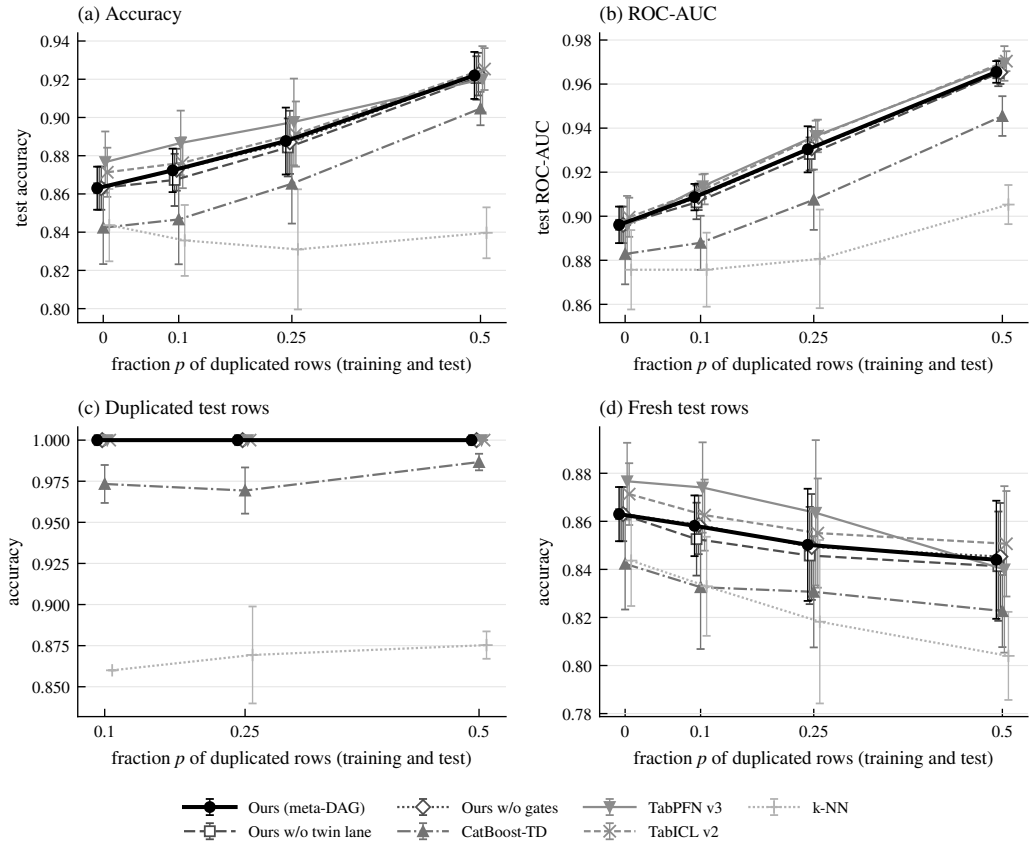


Figure 11: Toy B, variant B2 (repeats in the development rows as well): the panels of Figure 10; the $p = 0$ point is variant B's.

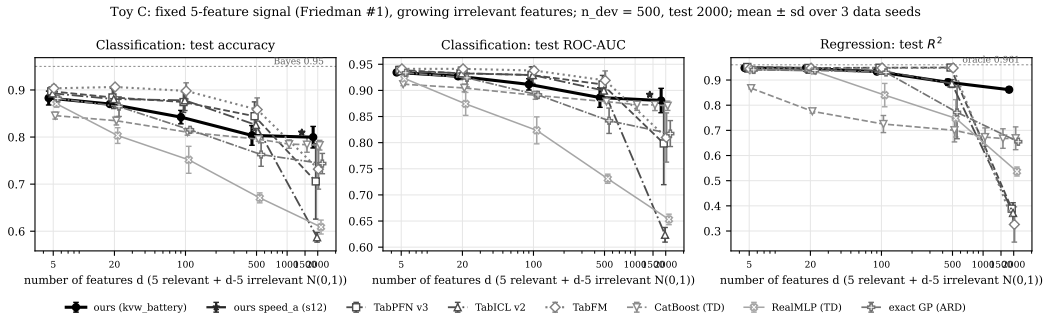


Figure 12: Toy C: accuracy, ROC-AUC and R^2 against the number of features d (5 relevant, the rest irrelevant; mean and standard deviation over 3 data sets).

sharply; the positive slopes of the pre-registered analysis come from that last point. With five values of d the location of the change is known only to lie in $(500, 2000]$.

- **Cost.** Ours takes about 5–130× longer than the foundation models: mean fit-plus-predict 137 s → 1,916 s (classification) and 133 s → 1,014 s (regression) from $d = 5$ to 2,000, against 7–124 s for the foundation models, on shared GPUs.
- **Scope.** One generating function, $n = 500$, 3 data sets: an illustration of the mechanism behind the pre-registered high-dimensional hypothesis, not a replacement for the high-dimensional benchmark group.

Table 30: Toy C, classification: accuracy (mean and standard deviation over 3 data sets) and mean ROC-AUC against the number of features d (5 relevant). Bayes accuracy 0.95. Bold: best in the column. Pre-registered; 105 cells.

accuracy	$d = 5$	20	100	500	2000
ours	$0.883_{\pm.015}$	$0.870_{\pm.001}$	$0.842_{\pm.014}$	$0.803_{\pm.021}$	$\mathbf{0.800}_{\pm.023}$
TabPFN v3	$0.897_{\pm.009}$	$0.884_{\pm.019}$	$0.874_{\pm.022}$	$0.844_{\pm.030}$	$0.706_{\pm.080}$
TabICLv2	$0.893_{\pm.005}$	$0.881_{\pm.020}$	$0.878_{\pm.022}$	$0.826_{\pm.026}$	$0.587_{\pm.010}$
TabFM	$\mathbf{0.903}_{\pm.007}$	$\mathbf{0.906}_{\pm.003}$	$\mathbf{0.898}_{\pm.017}$	$\mathbf{0.858}_{\pm.025}$	$0.732_{\pm.043}$
CatBoost-TD	$0.845_{\pm.007}$	$0.835_{\pm.009}$	$0.810_{\pm.006}$	$0.795_{\pm.004}$	$0.783_{\pm.010}$
RealMLP-TD	$0.871_{\pm.007}$	$0.803_{\pm.017}$	$0.752_{\pm.029}$	$0.671_{\pm.011}$	$0.609_{\pm.015}$
exact GP (ARD)	$0.882_{\pm.005}$	$0.865_{\pm.021}$	$0.815_{\pm.003}$	$0.763_{\pm.025}$	$0.744_{\pm.022}$
ROC-AUC					
ours	0.934	0.927	0.912	0.886	**0.881**
TabPFN v3	0.939	0.933	0.930	0.911	0.798
TabICLv2	0.937	0.933	0.929	0.900	0.624
TabFM	**0.941**	**0.941**	**0.938**	**0.919**	0.810
CatBoost-TD	0.912	0.905	0.891	0.878	0.871
RealMLP-TD	0.923	0.874	0.824	0.731	0.653
exact GP (ARD)	0.934	0.924	0.891	0.842	0.817

Table 31: Toy C, regression: test R^2 (mean and standard deviation over 3 data sets) against the number of features d (5 relevant). Mean test-oracle R^2 0.961. Bold: best in the column. Pre-registered; 105 cells.

R^2	$d = 5$	20	100	500	2000
ours	$0.948_{\pm.003}$	$0.946_{\pm.003}$	$0.934_{\pm.009}$	$0.891_{\pm.012}$	$\mathbf{0.862}_{\pm.009}$
TabPFN v3	$\mathbf{0.952}_{\pm.002}$	$\mathbf{0.948}_{\pm.003}$	$\mathbf{0.949}_{\pm.003}$	$\mathbf{0.950}_{\pm.003}$	$0.389_{\pm.004}$
TabICLv2	$0.946_{\pm.001}$	$0.941_{\pm.004}$	$0.937_{\pm.005}$	$0.886_{\pm.016}$	$0.372_{\pm.041}$
TabFM	$0.945_{\pm.004}$	$0.947_{\pm.002}$	$0.948_{\pm.003}$	$0.948_{\pm.002}$	$0.326_{\pm.070}$
CatBoost-TD	$0.868_{\pm.003}$	$0.776_{\pm.007}$	$0.725_{\pm.034}$	$0.700_{\pm.046}$	$0.669_{\pm.045}$
RealMLP-TD	$0.941_{\pm.002}$	$0.936_{\pm.002}$	$0.840_{\pm.046}$	$0.748_{\pm.074}$	$0.536_{\pm.018}$
exact GP (ARD)	$0.940_{\pm.004}$	$0.938_{\pm.003}$	$0.930_{\pm.002}$	$0.771_{\pm.105}$	$0.655_{\pm.002}$

G.4 TOY D: THE NUMBER OF ROWS

Design (pre-registered). 20 features $x \sim \mathcal{N}(0, 1)$, all relevant: f is the sum of four Friedman #1 functions on disjoint blocks of 5 columns. Regression $y = f + \mathcal{N}(0, 2^2)$ (test-oracle R^2 0.963); classification $y = 1[f > \text{median}]$ with 5 % label flips (Bayes accuracy 0.9465). One pool of 8,000 development and 2,000 test rows; $n \in \{500, 1000, 2000, 4000, 8000\}$ are nested prefixes of the development pool, and the test rows are the same for every n. Models: ours (no row cap), TabPFN v3, TabICLv2, CatBoost-TD (CPU), RealMLP-TD; 50 cells. The fixed analysis: OLS slope of $\log_{10}$ fit-plus-predict seconds on $\log_{10} n$ over all five points and over $n \geq 2000$, and score against n.

Results (Tables 33 and 34, Figure 14).

- **Regression.** Ours is the best model at every $n \leq 4000$ (e.g. 0.847 against 0.845 and 0.831 at $n = 500$) and second at $n = 8000$, 0.004 behind RealMLP-TD and level with TabPFN v3.
- **Classification.** Ours is 0.3 to 2.8 points below TabPFN v3 and TabICLv2 at every n, below RealMLP-TD from $n = 2000$, and above CatBoost-TD at every n.
- **Learning curve.** Ours improves steadily with n (accuracy 0.796 $\rightarrow$ 0.869, R^2 0.847 $\rightarrow$ 0.947) and fits all 8,000 rows with the exact GPs.
- **Cost.** At $n = 8000$ ours takes about 1,490 s per fit and prediction, about 80–110$\times$ TabPFN v3 and 27–40$\times$ RealMLP-TD. Over $n \geq 2000$ its time grows roughly linearly (slopes 0.76–1.02), well below the cubic cost of an exact GP solve: fixed per-lane costs (hyperparameter searches, the CatBoost and MLP lanes) dominate at these sizes. The near-zero or negative slopes of the foundation models and CatBoost reflect fixed overheads (checkpoint loading is timed) and contention, not scaling exponents.

Table 32: Toy C. Left: change of each model's score from $d = 5$, in points (mean over 3 data sets). Right: the pre-registered H1 analogue, gap = ours − best foundation model per table in points (mean over data sets; at $d = 2000$ the standard deviation in parentheses), the OLS slope of the gap on $\log_{10} d$ for each data set, and the verdict under the plan's rule.

	accuracy		R^2	
model	$d = 500$	2000	500	2000
ours	−7.9	**−8.3**	−5.7	**−8.6**
TabPFN v3	−5.3	−19.1	−0.2	−56.3
TabICLv2	−6.7	−30.6	−6.0	−57.4
TabFM	−4.5	−17.2	+0.3	−61.9
CatBoost-TD	−5.0	−6.3	−16.7	−19.9
RealMLP-TD	−20.0	−26.2	−19.3	−40.5
exact GP (ARD)	−11.9	−13.8	−16.9	−28.6

gap (points)	$d = 5$	20	100	500	2000	slopes per data set	verdict
accuracy (primary)	−2.07	−3.62	−5.63	−5.48	+4.05 (2.18)	+1.67, +2.09, +0.82	supported
ROC-AUC	−0.68	−1.41	−2.73	−3.29	+3.28 (4.06)	+1.85, +1.32, −0.52	mixed
R^2	−0.39	−0.22	−1.54	−5.86	+46.81 (1.12)	+13.4, +13.6, +12.6	supported

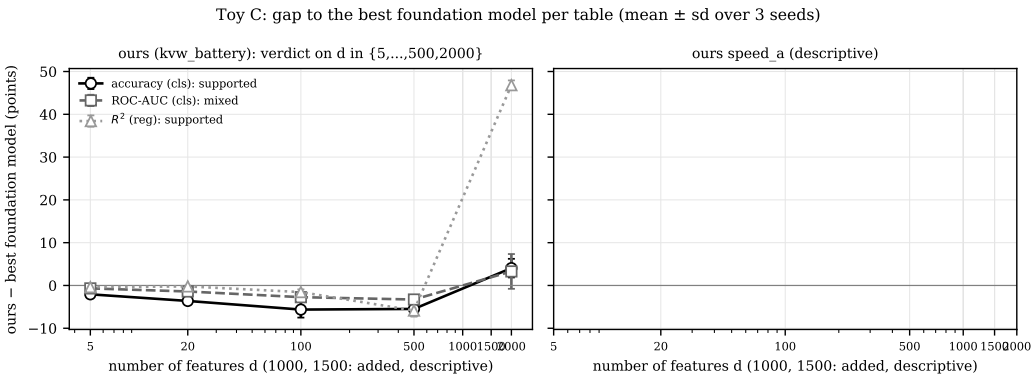


Figure 13: Toy C: gap between ours and the best foundation model per table (points; mean and standard deviation over 3 data sets) for accuracy, ROC-AUC and R^2; the pre-registered verdicts use $d \in \{5, 20, 100, 500, 2000\}$.

- **Timings are not clean.** The shared GPU pool was saturated (mean utilisation over the 8 GPUs 43–88 % at the start of each Toy D GPU job; the job's own GPU at 97–100 % in 28 of 40 cases); read the times as order-of-magnitude comparisons.

H Additional analyses

This section collects the analyses beyond the main comparison. Table 35 lists them with their status: *pre-registered* means the design and the analysis were written into a dated amendment of the benchmark protocol (Table 5) before any of its results existed; *exploratory* means the analysis was decided after results had been seen and is reported as descriptive only. All of them use saved test predictions or refits under the benchmark's fitting rules (one fit per cell, seed 0, all dev rows, no test rows or labels at fit time), and no result of this section was used to change our method.

H.1 Complementarity with the foundation models

Design (pre-registered, Amendment 8). For every foundation model $F \in \{$TabPFN v3, TabICLv2, TabFM, TabPFN v2.5$\}$ and every partner $P \in \{$ours, each other of these foundation models, CatBoost-TD (bag 5), RealMLP-TD$\}$, the ensemble $F{+}P$ is the equal-weight average of the two saved test predictions (class probabilities, or regression predictions; the weights $1/2$ were fixed in the amendment). Nothing is fitted or tuned. Per group we report the paired gain score$(F{+}P)-$score(F)

Table 33: Toy D: fit-plus-predict seconds against the number of development rows n, and the pre-registered OLS slopes of $\log_{10}$ time on $\log_{10} n$ over all five points and over $n \geq 2000$ (with standard errors for ours). Contended shared GPUs; CatBoost-TD on 8 CPU threads. 50 cells.

	500	1000	2000	4000	8000	slope, all	slope, $n \geq 2000$
classification							
ours	380	293	363	594	1489	0.50 ± 0.19	1.02 ± 0.18
TabPFN v3	19	13	11	9	14	-0.15	0.13
TabICLv2	19	11	7	6	11	-0.25	0.31
CatBoost-TD	2	3	10	12	11	0.72	0.03
RealMLP-TD	9	16	14	18	56	0.56	0.98
regression							
ours	259	299	514	368	1483	0.53 ± 0.20	0.76 ± 0.72
TabPFN v3	17	15	11	9	19	-0.05	0.37
TabICLv2	10	7	9	7	14	0.09	0.34
CatBoost-TD	31	31	24	20	15	-0.27	-0.33
RealMLP-TD	9	7	16	20	38	0.55	0.63

Table 34: Toy D: test accuracy (classification; Bayes 0.9465) and test R^2 (regression; oracle 0.963) against the number of development rows n. Bold: best in the column. Pre-registered; 50 cells.

	500	1000	2000	4000	8000
classification accuracy					
ours	0.796	0.818	0.826	0.860	0.869
TabPFN v3	0.810	0.838	0.848	**0.870**	0.886
TabICLv2	**0.816**	**0.846**	**0.849**	0.866	0.872
CatBoost-TD	0.783	0.792	0.810	0.843	0.860
RealMLP-TD	0.741	0.796	0.841	**0.870**	**0.887**
regression R^2					
ours	**0.847**	**0.908**	**0.933**	**0.943**	0.947
TabPFN v3	0.831	0.896	0.919	0.941	0.946
TabICLv2	0.845	0.880	0.896	0.923	0.933
CatBoost-TD	0.422	0.607	0.748	0.844	0.910
RealMLP-TD	0.724	0.871	0.913	0.937	**0.951**

in points of the primary metric, its 95% percentile bootstrap interval over tables (10,000 resamples), wins/ties/losses, and a two-sided Wilcoxon signed-rank test with Holm correction over the partners of one F. Error diversity is measured per pair on the fresh tables as the Pearson correlation of per-row log-loss and the share of test rows on which the two predicted classes differ, averaged over tables.

Results. Table 36 gives the gain of adding our method to each foundation model. On the fresh tables every interval covers zero (-0.12 to $+0.07$ points). On the high-dimensional tables, TabICLv2 + ours gains $+1.17$ points $[+0.15, +2.37]$ (7/2/3) and TabPFN v3 + ours $+0.34$ $[-0.05, +0.83]$ (7/3/2); neither survives the Holm correction over the partners. Over all 51 TabArena tables the one significant effect of our method as a partner is on TabFM (-0.23 $[-0.38, -0.08]$, Holm $p = .003$), and averaging with TabFM gives the largest gain to each of the other three foundation models. Our method's errors are less correlated with those of the foundation models than the foundation models' errors with each other (Figure 15): on the fresh tables the per-row log-loss correlation of ours with each foundation model is 0.87–0.90 (21 tables), against 0.95–0.97 between any two foundation models (26 tables), and the predicted class differs on 3.0–3.4% of test rows, against 1.3–1.7% between foundation models.

H.2 Distribution shift along the first principal component

Design (pre-registered, Amendment 9). The tables are every fresh and held-out table with at most 4,000 dev rows and at least two numeric (non-categorical) features: 14 tables (10 fresh, 4 held-out; Table 37); hiva-agnostic and splice are excluded because they have no numeric feature. Per table the dev and test rows are pooled, the numeric features are z-scored over all pooled rows

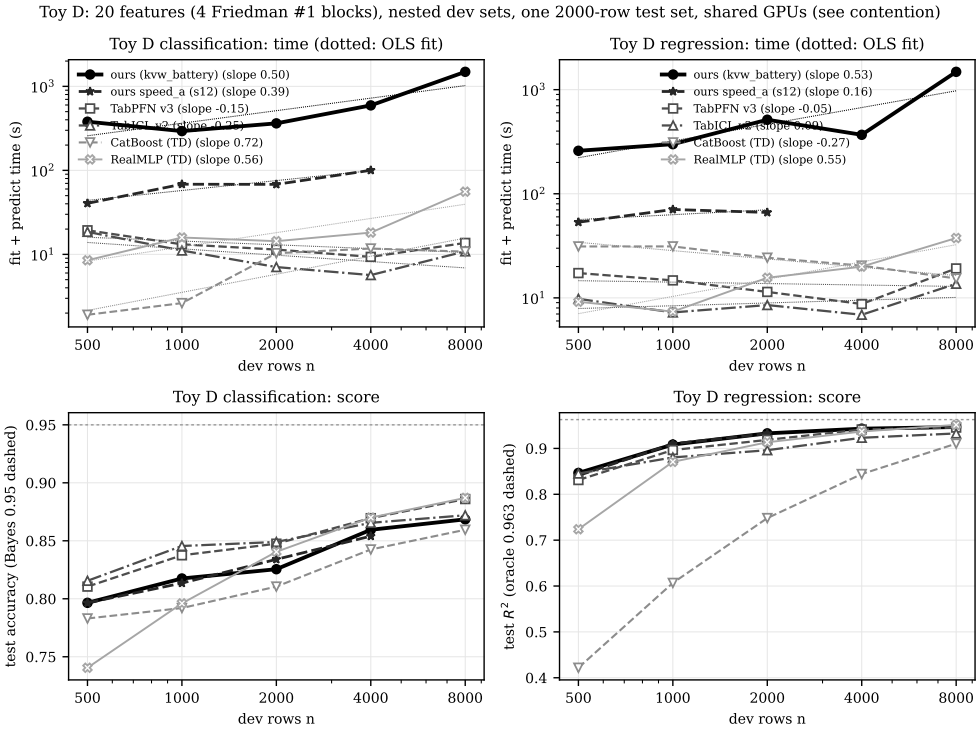


Figure 14: Toy D: fit-plus-predict time (top, log–log with OLS fits) and score (bottom) against the number of development rows, for classification (left) and regression (right). Times were measured on shared, contended GPUs.

Table 35: Additional analyses. Status: pre-registered (the protocol amendment that fixed the design before any result) or exploratory. "Ours" is the variant of our method the analysis uses: TabEvo is the evolved program as selected.

analysis	§	status	ours	data
complementarity	H.1	pre-reg. (Am. 8)	TabEvo[a]	71 tables
distribution shift	H.2	pre-reg. (Am. 9)	TabEvo	14 tables × 2 splits
when to use	H.3	exploratory	TabEvo	71 tables
feature relevance	H.5.1	pre-reg. (Am. 12)	TabEvo	24 Toy C tables
battery head weights	H.5.2	exploratory	seed and final program	15 battery tables
learning curves	H.6	pre-reg. (Am. 11)	TabEvo	11 tables × 4 fractions

[a] In the full-data regime the analysis takes the full-data variant of TabEvo (§C.4) where it has a finished cell (2 tables) and TabEvo elsewhere.

(no labels are used) and their first principal component PC1 is computed. The *shift* split puts the $n - \lceil 0.3n \rceil$ rows with the lowest PC1 score into dev and the rest into test, so the test rows lie beyond the dev rows along the main axis of variation. The *iid* control is a random split of the same pooled rows with the same sizes (seed 0, stratified on classification). Models: TabEvo, TabPFN v3, TabICLv2, TabFM, CatBoost-TD (bag 5), LightGBM-TD (bag 5) and RealMLP-TD (standardized inputs), under the benchmark's fitting rules. The analysis fixed in the amendment: per model, the mean drop from the iid to the shift split (points, with a bootstrap interval over tables) and the relative drop; ours against each model on the shift split (W/T/L, Wilcoxon with Holm correction). A failed cell is scored as the worst model on its (table, split), as in the main benchmark.

Status. All 28 cells (14 tables × 2 splits) of each of the six comparators finished. Of the 28 cells of TabEvo, 21 finished and one failed (anneal, shift split); the rest are missing, so the pre-registered comparison is not reported.

Table 36: Complementarity (pre-registered, Amendment 8): gain of the equal-weight ensemble F+ours over the foundation model F alone, in points of the primary metric (full data). n: tables where both members have a finished cell; CI: 95% bootstrap interval over tables; W/T/L: tables where F+ours is better / equal / worse than F; p: two-sided Wilcoxon, Holm-corrected over the partners of F. Last column: the other partner with the largest mean gain for this F and group, on its own tables (n in parentheses). TabPFN v2.5 was not run on the high-dimensional group.

F	group	n	$\bar{\Delta}$	95% CI	W/T/L	p_{Holm}	best other P ($\bar{\Delta}$, n)
TabPFN v3	fresh	21	-0.05	$[-0.20, +0.11]$	7/5/9	1	TabFM ($+0.20$, 26)
	high-dim.	12	$+0.34$	$[-0.05, +0.83]$	7/3/2	.445	TabFM ($+0.76$, 16)
	TabArena	46	-0.12	$[-0.26, +0.02]$	15/7/24	.350	TabFM ($+0.34$, 51)
TabICLv2	fresh	21	$+0.07$	$[-0.10, +0.25]$	7/5/9	1	RealMLP-TD ($+0.25$, 11)
	high-dim.	12	$+1.17$	$[+0.15, +2.37]$	7/2/3	.324	TabFM ($+1.49$, 16)
	TabArena	46	$+0.10$	$[-0.08, +0.30]$	18/7/21	1	TabFM ($+0.44$, 51)
TabFM	fresh	21	-0.12	$[-0.25, +0.03]$	5/2/14	.571	TabPFN v2.5 ($+0.01$, 26)
	high-dim.	12	-0.04	$[-0.47, +0.52]$	4/2/6	.400	TabPFN v3 (-0.14, 16)
	TabArena	46	-0.23	$[-0.38, -0.08]$	9/4/33	.003	TabPFN v3 (-0.02, 51)
TabPFN v2.5	fresh	21	-0.10	$[-0.22, +0.01]$	5/2/14	.335	TabFM ($+0.27$, 26)
	TabArena	46	$+0.06$	$[-0.13, +0.33]$	17/5/24	.717	TabFM ($+0.67$, 51)

TabArena: all 51 TabArena tables (fresh, held-out and battery pooled).

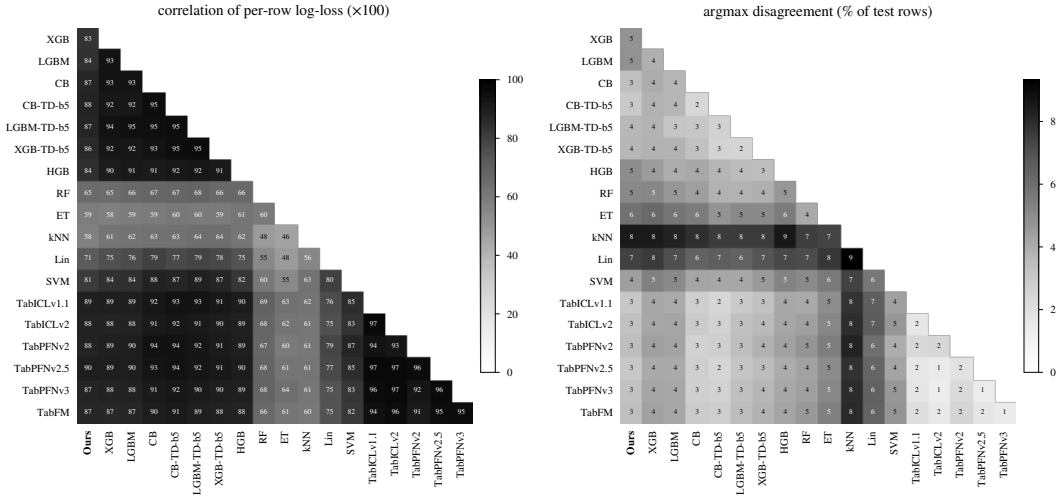


Figure 15: Error diversity on the fresh tables (full data; complementarity analysis, pre-registered, Amendment 8). Left: Pearson correlation ($\times 100$) of the per-row log-loss of two models; right: share of test rows (%) on which their predicted classes differ. Each entry is the mean over the fresh tables where both models have a finished cell (21 for pairs with ours, 26 otherwise).

H.3 When to use our method: table properties and outcomes

Design (exploratory, not pre-registered). From the saved test predictions of the full benchmark and the data alone (no fit), four per-table outcomes in points of the primary metric (positive: ours better): ours minus the best foundation model on that table (the best of TabFM, TabPFN v3, TabPFN v2.5, TabICLv2), ours minus TabICLv2, ours minus the best trained baseline (the best of CatBoost-, LightGBM- and XGBoost-TD bag 5, RealMLP-TD and TabM-D), and ours minus CatBoost-TD (bag 5). A table enters an outcome only where ours and the comparators have finished cells. The outcomes are related to 14 table properties computed on the dev rows: n, d, n/d, categorical and binary feature fractions, number of classes, minority-class share, the share of test rows that repeat a dev row, the share of duplicated dev rows, target skewness (regression), mean absolute feature correlation, number of principal components for 90% of the variance, and missing-cell fraction. Per scope we report Spearman correlations with Benjamini–Hochberg correction over the 4×14 family, and a depth-2 regression tree per outcome (at least 5 tables per leaf) together with two checks against over-reading it: its leave-one-out R^2 and a 1,000-permutation test of its in-sample R^2 in which the whole tree search is repeated. Scopes: the four groups; *all* (51 TabArena + 20

Table 37: The 14 tables of the distribution-shift analysis (pre-registered, Amendment 9). Pooled: dev + test rows of the benchmark split; numeric: the non-categorical features, all of which enter the principal component analysis; PC1: its share of the variance of the z-scored numeric features. Both splits have the same sizes (dev / test) and were fixed before any result.

table	group	task	pooled	features	numeric	PC1 var.	dev / test
anneal	fresh	multiclass (5)	898	38	6	0.219	628 / 270
bioresponse	fresh	binary	3,751	1,776	1,776	0.108	2,625 / 1,126
churn	fresh	binary	5,000	19	15	0.138	3,500 / 1,500
credit-g	fresh	binary	1,000	20	7	0.237	700 / 300
fitness-club	fresh	binary	1,500	6	3	0.487	1,050 / 450
is-this-a-good-customer	fresh	binary	1,723	13	5	0.331	1,206 / 517
marketing-campaign	fresh	binary	2,240	25	16	0.379	1,568 / 672
mic	fresh	multiclass (8)	1,699	111	17	0.138	1,189 / 510
polish-companies-bankruptcy	fresh	binary	5,910	64	64	0.178	4,137 / 1,773
seismic-bumps	fresh	binary	2,584	15	11	0.325	1,808 / 776
used-fiat-500[b]	held-out	regression	1,538	7	6	0.341	1,076 / 462
diabetes	held-out	binary	768	8	8	0.262	537 / 231
hazelnut-contaminant[b]	held-out	binary	2,400	30	30	0.432	1,680 / 720
qsar-tid-11	held-out	regression	5,742	1,024	1,024	0.021	4,019 / 1,723

[b] another-dataset-on-used-fiat-500 and hazelnut-spread-contaminant-detection.

high-dimensional tables, pooled); *unseen* (fresh + held-out + high-dim., i.e. without the battery); *all rows* (all tables where our method fitted every dev row).

Findings. Against the trained baselines our method is ahead on numeric, small to mid-sized tables: ours minus CatBoost-TD (bag 5) is $+0.75$ points over the 58 tables (37/4/17, Table 38), $+1.55$ on the battery, $+1.48$ on the high-dimensional tables and $+0.01$ on the fresh tables. The difference to the trained baselines decreases with the categorical fraction (ours minus best trained: $\rho = -0.42$, BH $p = .028$; Table 39) and the difference to CatBoost-TD (bag 5) with the number of dev rows ($\rho = -0.45$, BH $p = .021$; on the 47 tables fitted on all rows $\rho = -0.30$, unadjusted $p = .044$). Against TabICLv2, the one table property with predictive value out of sample is the number of features: the tree splits at $d > 997$, where ours is ahead on all 6 tables (mean $+4.01$ points; bioresponse, qsar-tid-11, arcene and the three QSAR-TID tables of the high-dimensional group), and it is the only tree with a clearly positive leave-one-out R^2 (0.29 on all tables, 0.50 on the unseen ones; permutation $p \leq .003$; Table 40). On five of these six tables TabFM is above ours (all but arcene). Against the best foundation model the difference is negative in every scope (fresh -0.79, all -1.30 points), and no table property predicts it out of sample (leave-one-out R^2 at most 0.01).

H.4 High-dimensional tables

The high-dimensional group is the pre-registered test set of hypothesis H1 (Amendment 2; §A.4): 20 OpenML tables with 122 to 10,935 features, listed in Table 4. Every method's score on each of them is in the second half of the per-table tables (§I.3), the pooled statistics are in Table 48 and the critical-difference diagram in Figure 25. The analysis fixed in the amendment (the per-table difference to the best foundation model; wins, ties and losses and Holm-corrected Wilcoxon tests against each foundation model on the group; the OLS slope of the difference on $\log_{10} d$ over all tables) is computed by the released aggregation code and drawn in Figure 16. H1 is not supported: five of the program's exact-GP fits exceed the time limit and one fails numerically (Appendix I), the best foundation model is ahead of it on the tables it completes (1/1/10 on the 12 tables of Table 38), and the OLS slope of the difference on $\log_{10} d$ is negative, so the hypothesised margin that grows with the number of features does not appear.

Two further results locate where the program's advantage lies. On the tables where both have finished cells, the program is ahead of TabICLv2 on 7 of 12 high-dimensional tables ($+1.08$ points on average) and of bagged CatBoost-TD on 8 of 12 ($+1.48$), while the best foundation model is ahead of it on 10 (Table 38). Among all table properties, the number of features is the one that predicts the program's margin over TabICLv2 out of sample: on each of the 6 tables with more than 997 features the program is the more accurate of the two, by $+4.01$ points on average (Table 40; exploratory).

Table 38: Per-table outcomes (exploratory): ours minus comparator in points of the primary metric, full data, complete case (tables where ours and the comparator have finished cells). Best FM / best trained: the best foundation model / trained baseline on that table. W/T/L from ours' side; p: two-sided Wilcoxon signed-rank (zero differences split), unadjusted. Ours is TabEvo.

ours minus	scope	n	mean	median	W/T/L	p
best FM	fresh	21	-0.79	-0.53	0/2/19	$<.001$
	held-out	10	-1.65	-1.29	1/1/8	.031
	battery	15	-1.70	-1.36	0/1/14	$<.001$
	high-dim.	12	-1.42	-1.48	1/1/10	.023
	all	58	-1.30	-1.00	2/5/51	$<.001$
	all rows	47	-1.11	-0.96	2/4/41	$<.001$
TabICLv2	fresh	21	-0.31	-0.33	5/1/15	.071
	held-out	10	-0.08	-0.49	3/0/7	.492
	battery	15	-1.10	-0.69	3/0/12	.012
	high-dim.	12	$+1.08$	$+0.13$	7/0/5	.569
	all	58	-0.19	-0.38	18/1/39	.022
	all rows	47	$+0.07$	-0.33	17/1/29	.210
best trained	fresh	21	-0.21	$+0.00$	7/4/10	.169
	held-out	10	-0.18	$+0.05$	5/0/5	1.000
	battery	15	$+0.59$	$+0.46$	12/0/3	.026
	high-dim.	12	$+0.88$	$+0.39$	8/2/2	.057
	all	58	$+0.23$	$+0.14$	32/6/20	.099
	all rows	47	$+0.38$	$+0.22$	30/6/11	.008
CatBoost-TD (bag 5)	fresh	21	$+0.01$	$+0.05$	11/2/8	.509
	held-out	10	$+0.25$	$+0.05$	5/0/5	.695
	battery	15	$+1.55$	$+2.05$	13/0/2	.001
	high-dim.	12	$+1.48$	$+1.02$	8/2/2	.023
	all	58	$+0.75$	$+0.40$	37/4/17	$<.001$
	all rows	47	$+1.00$	$+0.54$	35/4/8	$<.001$

Table 39: Spearman correlation of each outcome with each table property (exploratory), all 71 tables pooled (51 TabArena + 20 high-dimensional); $n = 58$ tables with a finished cell of ours ($n = 39$ for the two class properties, 19 for target skewness, 55 for the two correlation properties). *: Benjamini–Hochberg-adjusted $p < 0.05$ over the 4×14 family; †: unadjusted $p < 0.05$ only.

	ours minus			
table property	best FM	TabICLv2	best trained	CatBoost-TD (bag 5)
n (dev rows)	-0.08	-0.09	$-0.26^\dagger$	-0.45^*
d (features)	$+0.07$	$+0.28^\dagger$	-0.05	-0.07
n/d	-0.02	$-0.26^\dagger$	-0.17	$-0.28^\dagger$
categorical fraction	$+0.14$	-0.05	-0.42^*	$-0.34^\dagger$
binary fraction	$+0.11$	$+0.34^\dagger$	-0.23	-0.14
number of classes	-0.31	-0.27	$+0.03$	$+0.10$
minority-class share	-0.08	$+0.08$	$+0.29$	$+0.22$
test rows repeating a dev row	-0.07	-0.02	-0.05	$+0.05$
duplicated dev rows	-0.10	-0.05	-0.07	$+0.03$
target skewness	$+0.20$	-0.13	-0.08	$+0.05$
mean $\lvert r \rvert$ between features	-0.03	$-0.29^\dagger$	$+0.02$	$+0.09$
components for 90% variance	-0.07	$+0.22$	$+0.07$	$+0.03$
missing-cell fraction	-0.11	$-0.29^\dagger$	-0.25	$-0.26^\dagger$
regression task	$-0.35^\dagger$	-0.07	$+0.29^\dagger$	$+0.26$

The controlled version of this effect is Toy C (Appendix G): with 5 relevant among 2,000 features the program is the most accurate model of the study, while the foundation models degrade.

Table 40: Depth-2 regression trees of each outcome on the 14 table properties (exploratory; at least 5 tables per leaf): in-sample R^2, leave-one-out (LOO) R^2, median in-sample R^2 over 1,000 permutations of the outcome with the tree search repeated, and the permutation p. Only the TabICLv2 trees keep a clearly positive LOO R^2; their first split is at $d = 997$ features.

scope	ours minus	n	R^2	LOO R^2	perm. median	p_{perm}
all	best FM	58	0.49	$+0.01$	0.25	.011
	TabICLv2	58	0.57	$+0.29$	0.26	.001
	best trained	58	0.32	-0.07	0.24	.133
	CatBoost-TD (bag 5)	58	0.42	-0.01	0.26	.024
unseen	best FM	43	0.27	-0.20	0.31	.707
	TabICLv2	43	0.68	$+0.50$	0.31	.002
	best trained	43	0.34	-0.06	0.26	.117
	CatBoost-TD (bag 5)	43	0.48	-0.45	0.29	.021
all rows	best FM	47	0.35	-0.32	0.30	.290
	TabICLv2	47	0.66	$+0.45$	0.31	.003
	best trained	47	0.35	-0.15	0.25	.111
	CatBoost-TD (bag 5)	47	0.51	-0.12	0.30	.034

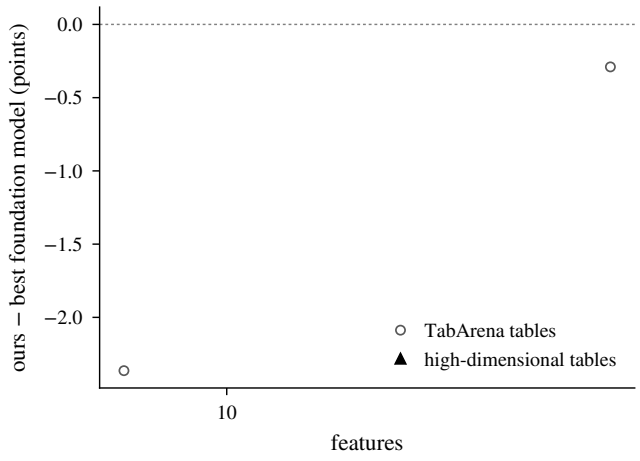


Figure 16: Pre-registered H1 analysis (Amendment 2): per-table difference between TabEvo and the best foundation model (primary metric, points) against the number of features; open circles: TabArena tables, filled triangles: high-dimensional tables; the line is the OLS fit on $\log_{10} d$ with its 95 % band.

H.5 Interpretability

H.5.1 Recovery of the relevant features (Toy C)

Design (pre-registered, Amendment 12). The Toy C tables of Appendix G have a known ground truth: the Friedman #1 function of 5 relevant features plus $d - 5$ irrelevant $N(0, 1)$ features, 500 dev rows, $d \in \{20, 100, 500, 2000\}$, classification and regression, three data seeds (24 tables; at $d = 5$ no feature is irrelevant). TabEvo is refitted exactly as in the benchmark (seed 0, all dev rows), and a per-feature relevance vector is read from two of its learned components, both fixed in the amendment: the inverse lengthscales s_j of the Kumaraswamy-warp edge `front_Wk`, and the column norms of the learned linear metric L of edge `metric` ($z = Lx$). Both edges read the table directly, so their fitted parameters are those of the full fit; this was checked on two tables, where the parameters and the test predictions equal those of the full fit and of the benchmark cell exactly. (The four GP prediction edges have one lengthscale each, so there is no per-feature lengthscale of the GP on the raw row.) Comparators, also fixed in the amendment: CatBoost-TD's LossFunctionChange importance of the benchmarked fit, computed on the dev rows, and the permutation importance of TabPFN v3, cross-fitted over 5 folds of the dev rows (the test split is never used). Each relevance vector is scored by the ROC-AUC of separating the 5 relevant features from the irrelevant ones (1 = perfect, 0.5 = chance) and by precision at 5.

Results. Over the 24 tables the mean ROC-AUC is 0.992 for CatBoost-TD's loss-function-change importance, 0.974 for TabPFN v3's permutation importance, 0.940 for the Kumaraswamy inverse lengthscales of TabEvo and 0.807 for the column norms of its metric L (Table 41, Figure 17; per-table wins/ties/losses against the Kumaraswamy vector: CatBoost 16/7/1, Wilcoxon $p = .001$; TabPFN v3 15/7/2, $p = .005$; metric L 3/3/18, $p < .001$). The Kumaraswamy vector recovers four of the five relevant features at every d; its whole deficit is the term $20(u_3 - 0.5)^2$, a U-shaped function of x_3 that a monotone per-column warp cannot straighten. Without x_3 its mean AUC is 0.996: from $d = 100$ on classification and $d = 500$ on regression, x_3 ranks among the irrelevant columns (mean rank 24, 269 and 1,576 at $d = 100$, 500 and 2,000 on classification), while x_1, x_2, x_4 and x_5 keep mean ranks of 1–20. The relevance of TabEvo is read from parameters of the fitted model, with no computation beyond the fit; the permutation importance of TabPFN v3 needs 5 additional fits and $5(d + 1)$ predictions. Figure 18 shows both vectors on one table with $d = 100$.

Design note: the metric edge on wide tables. The column norms of L rank the relevant features above chance up to $d = 500$ and at chance at $d = 2000$ (AUC 0.49 and 0.47; the diagonal of L alone 0.500). This follows from the initialisation of the metric edge: L starts at the identity on z-scored columns, so with hundreds of columns the initial pairwise distances are large (median 32–63 on these tables) and the exponential kernel $\exp(-\|z - z'\|)$ is saturated (10^{-14} to 10^{-28}); the marginal-likelihood gradients fall below Adam's ϵ, and for $d \geq 500$ L stays within 10^{-4} of the identity, so the metric GP works in the standardised raw coordinates. The Kumaraswamy edge initialises its lengthscales at the median pairwise distance of a row subsample and learns at every d. Initialising the metric edge at the median distance is the natural refinement of the evolved program; it is not applied here, and every benchmark result uses the program as evolved.

Table 41: Feature-relevance recovery on the Toy C tables (pre-registered, Amendment 12): ROC-AUC of separating the 5 relevant from the $d - 5$ irrelevant features, mean $\pm$ sd over 3 data seeds, precision at 5 in brackets. TabEvo: inverse lengthscales of the Kumaraswamy-warp edge and column norms of the learned metric L. All 24 tables $\times$ 4 methods computed.

task	d	ours: Kumaraswamy s_j	ours: metric L	CatBoost-TD	TabPFN v3
classif.	20	0.991 ± 0.008 [0.87]	0.929 ± 0.056 [0.67]	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]
	100	0.950 ± 0.081 [0.80]	0.884 ± 0.011 [0.73]	1.000 ± 0.000 [1.00]	0.942 ± 0.100 [0.93]
	500	0.886 ± 0.093 [0.67]	0.939 ± 0.059 [0.73]	0.999 ± 0.000 [0.87]	0.999 ± 0.002 [0.93]
	2000	0.841 ± 0.022 [0.73]	0.488 ± 0.129 [0.00]	0.933 ± 0.115 [0.93]	0.851 ± 0.034 [0.67]
regression	20	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]
	100	0.999 ± 0.001 [0.87]	0.833 ± 0.022 [0.73]	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]
	500	0.953 ± 0.051 [0.87]	0.919 ± 0.095 [0.73]	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]
	2000	0.902 ± 0.032 [0.80]	0.466 ± 0.094 [0.00]	1.000 ± 0.000 [1.00]	1.000 ± 0.000 [1.00]

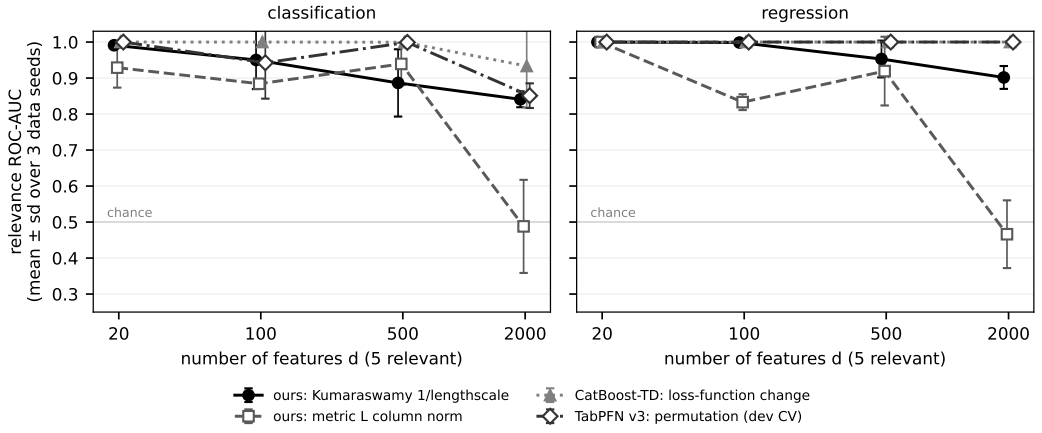


Figure 17: Feature-relevance recovery on the Toy C tables (pre-registered, Amendment 12): ROC-AUC of separating the 5 relevant features from the irrelevant ones against the number of features d, mean $\pm$ sd over 3 data seeds; 0.5 is chance.

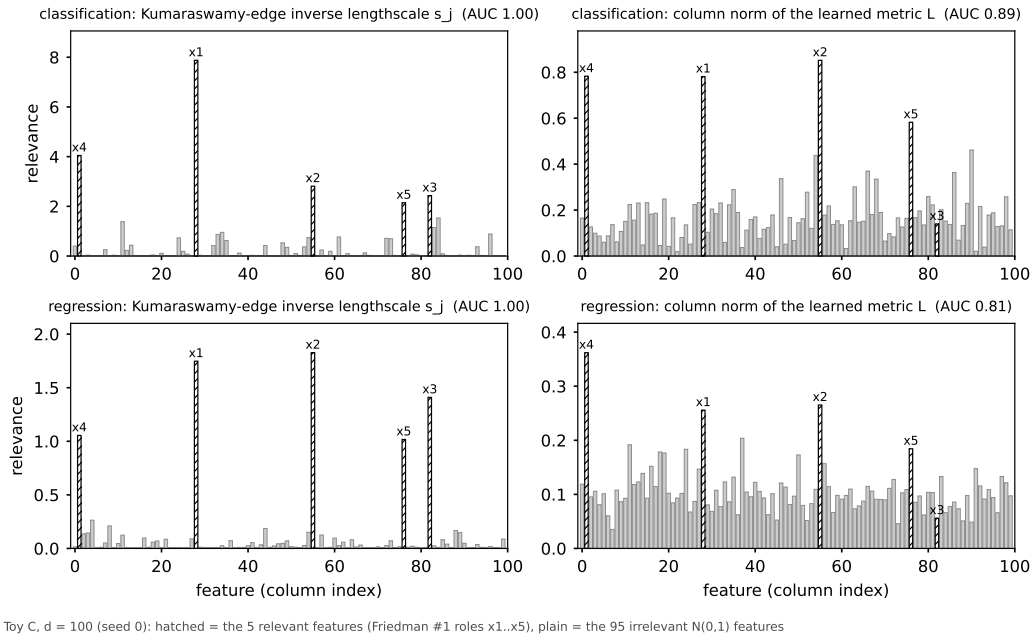


Figure 18: The two relevance vectors of TabEvo on one Toy C table with $d = 100$ (data seed 0), classification (top) and regression (bottom): hatched bars are the 5 relevant features (Friedman #1 roles $x_1, \ldots, x_5$), plain bars the 95 irrelevant ones. Left: inverse lengthscales s_j of the Kumaraswamy-warp edge; right: column norms of the learned metric L. ROC-AUC of each vector in the panel title.

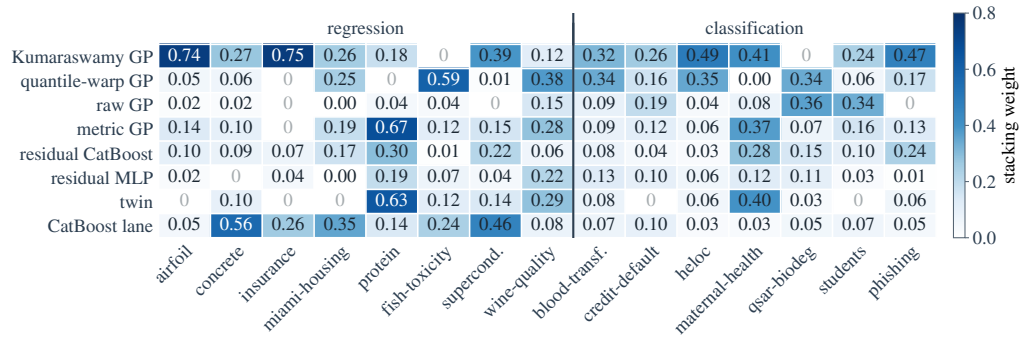


Figure 19: The program adapts its recipe to each table. Stacking weights of the eight blocks on the development tables (means over the cross-fitting folds). Each table settles on its own mixture: a warped GP dominates smooth regression targets, the gradient-boosted lane takes over where partitions help, the twin block where identical rows recur; no block is unused everywhere.

H.5.2 Head weights on the battery

Design (exploratory, not pre-registered). During the evolution run, the fitness evaluation stored, for every program, table and cross-validation fold, the weight the non-negative stacking head gave each prediction block. We read these stored weights for the seed (`71dd5eba`) and the final program (`20eb9f64`, i.e. TabEvo) on the 15 battery tables (3 folds each) and recover the stacking coefficient of each block (the stored value times $\sqrt{c}$, $c = \max(\#\text{classes}, 1)$). No model is refitted.

Results. Figure 20 shows the mean coefficient of every block on every table: the head uses different blocks on different tables. In the final program the GP on the Kumaraswamy-warped row carries the largest mean coefficient (0.33) and is the largest block on 6 of the 15 tables; the GP on the quantile-warped row is the largest on 3, the GP on the raw row on 2, and the metric-GP block on 1 (physiochemical-protein). The direct CatBoost lane `raw_forest` that evolution added is the largest block on 3 tables (concrete 0.56, superconductivity 0.46, miami-housing 0.35) and has a non-

zero coefficient in 43 of the 45 folds. Its mean coefficient is 0.27 on the 8 regression tables and 0.06 on the 7 classification tables, which matches where evolution's gain lies: across tables the coefficient and the per-table gain of the final program over the seed have Spearman $\rho = 0.71$ ($p = .047$, $n = 8$) on regression and $\rho = -0.04$ ($p = .94$, $n = 7$) on classification ($\rho = 0.38$, $p = .16$, and Pearson 0.71, $p = .003$, on all 15; Figure 21). The gain is positive on 11 of the 15 tables (mean $+0.12$ points).

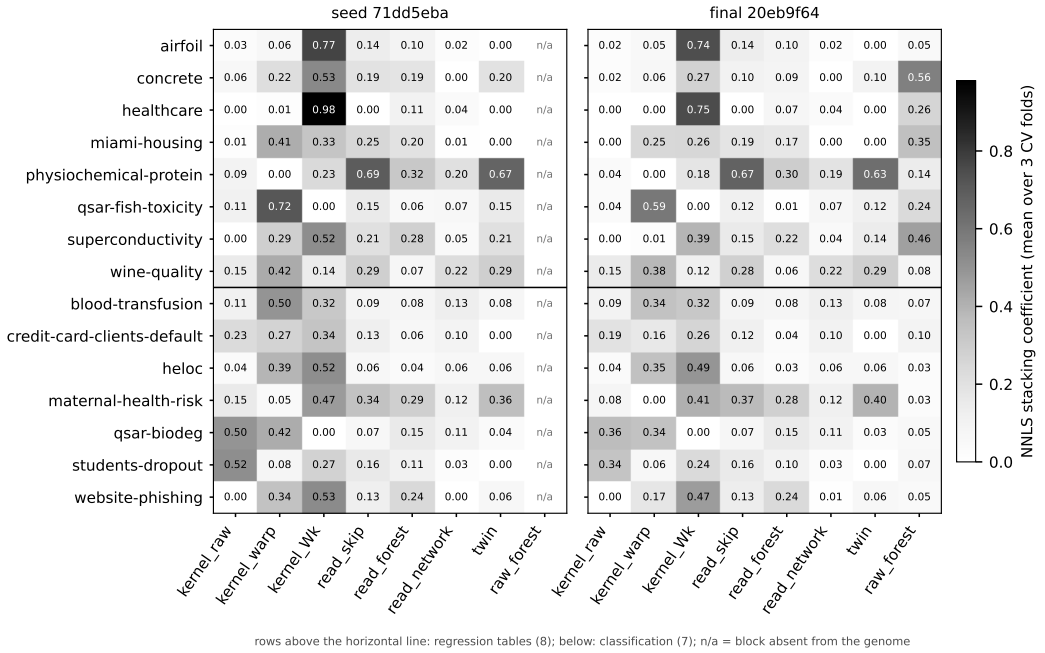


Figure 20: Non-negative stacking coefficient of each prediction block on each battery table, mean over the 3 cross-validation folds of the evolution run's fitness evaluation (exploratory). Left: the seed program; right: the final program (TabEvo), which adds the block `raw_forest` (n/a: absent from the seed). Rows above the line are the 8 regression tables, below the 7 classification tables.

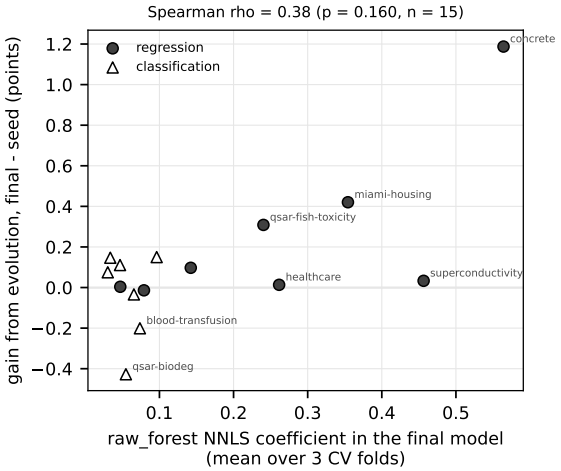


Figure 21: Per battery table (exploratory): the stacking coefficient of the evolved `raw_forest` block in the final program (mean over 3 folds) against the table's gain of the final program over the seed (battery fitness, points). Circles: regression; triangles: classification.

H.6 Learning curves

Design (pre-registered, Amendment 11). The 11 fresh tables with at most 4,000 dev rows (Table 43). For each fraction $f \in \{0.1, 0.25, 0.5\}$ a subsample of the dev rows is drawn (seed 0;

stratified with largest-remainder quotas on classification), nested so that the 0.1 subset lies inside the 0.25 subset, which lies inside the 0.5 subset; the test split is unchanged, and $f = 1$ is the benchmark cell. Models: TabEvo, TabPFN v3, TabICLv2, CatBoost-TD (bag 5) and RealMLP-TD (standardized inputs), under the benchmark's fitting rules. The analysis fixed in the amendment: mean primary score and average rank per fraction and model; ours minus each model per fraction (W/T/L, Wilcoxon with Holm correction over the four models within a fraction). A failed cell is scored as the worst of the five models on that (table, fraction); a complete-case variant keeps only the tables where all 20 cells finished.

Results. All 220 cells (11 tables $\times$ 4 fractions $\times$ 5 models) have a record: 213 succeeded and 7 of TabEvo's cells failed, all on the three multiclass tables (anneal and mic at $f = 0.1$ and 0.25, where the CatBoost lanes receive a constant training target on the smallest subsamples; hiva-agnostic at $f = 0.1$, 0.5 and 1, an exact-GP factorisation), and are scored as the worst of the five models on that (table, fraction), as pre-registered. Table 42 and Figure 22 give the pre-registered quantities. The gap between our method and the in-context models is largest with 10 % of the dev rows (a few hundred rows: -1.71 points to TabPFN v3, -1.17 to TabICLv2) and closes as rows are added (-0.35 and $+0.02$ at $f = 0.5$); from $f = 0.25$ on our method is level with bagged CatBoost-TD and ahead of RealMLP-TD, significantly so at $f = 0.25$ ($+1.71$, 8/3/0, Holm $p = .031$) and $f = 0.5$ ($+2.33$, 9/1/1, $p = .031$). On the 8 tables where all 20 cells succeeded, our method's average rank is 2.56, 2.75, 2.44 and 2.62 at the four fractions, and its difference to the best other model -0.49, -0.44, -0.14 and -0.38 points.

Table 42: Learning curves (pre-registered, Amendment 11): mean accuracy (points) and average rank per fraction f of the dev rows over the 11 fresh tables with at most 4,000 dev rows, failed cells scored as the worst of the five models (top); ours minus each model per fraction, with W/T/L from ours' side and the two-sided Wilcoxon p, Holm-corrected over the four models within a fraction (bottom); 220 cells.

	mean accuracy at f				average rank at f			
model	0.1	0.25	0.5	1	0.1	0.25	0.5	1
ours	84.51	87.12	88.52	89.19	3.09	3.00	2.73	2.95
TabPFN v3	86.21	87.77	88.87	89.56	2.00	2.05	2.00	2.27
TabICLv2	85.67	87.59	88.50	89.42	2.77	2.18	2.55	2.68
CatBoost-TD (bag 5)	86.19	87.17	88.33	89.11	2.91	3.05	3.14	3.18
RealMLP-TD	82.42	85.40	86.19	88.70	4.23	4.73	4.59	3.91

ours minus	$f = 0.1$	$f = 0.25$	$f = 0.5$	$f = 1$
TabPFN v3	-1.71 (3/1/7) .896	-0.65 (2/3/6) .609	-0.35 (3/2/6) 1	-0.37 (4/1/6) .668
TabICLv2	-1.17 (4/1/6) 1	-0.48 (3/2/6) .875	$+0.02$ (4/2/5) 1	-0.22 (3/1/7) .777
CatBoost-TD (bag 5)	-1.69 (4/2/5) 1	-0.06 (4/2/5) .875	$+0.19$ (6/1/4) 1	$+0.09$ (7/1/3) .777
RealMLP-TD	$+2.09$ (7/2/2) .188	$+1.71$ (8/3/0) .031	$+2.33$ (9/1/1) .031	$+0.50$ (6/2/3) .641

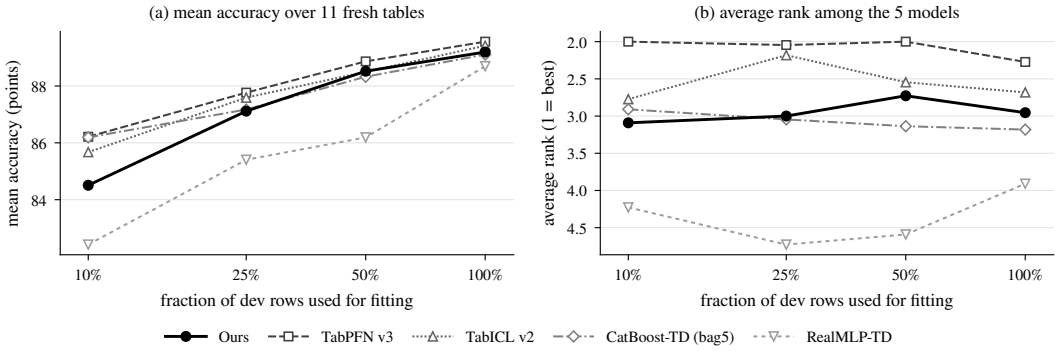


Figure 22: Learning curves (pre-registered, Amendment 11): mean accuracy (left) and average rank (right) against the fraction of dev rows, 11 fresh tables, failed cells scored as worst.

Table 43: Tables and dev rows of the learning-curve analysis (pre-registered, Amendment 11). The subsamples are nested ($f = 0.1 \subset 0.25 \subset 0.5 \subset 1$) and share the benchmark's test split; they were fixed before any result.

		dev rows at fraction f				
table	task	0.1	0.25	0.5	1	test rows
anneal	multiclass (5)	60	150	299	598	300
bioresponse	binary	250	625	1,250	2,500	1,251
churn	binary	333	833	1,667	3,333	1,667
credit-g	binary	67	167	333	666	334
fitness-club	binary	100	250	500	1,000	500
hiva-agnostic	multiclass (3)	256	641	1,282	2,563	1,282
is-this-a-good-customer	binary	115	287	574	1,148	575
marketing-campaign	binary	149	373	747	1,493	747
mic	multiclass (8)	113	283	566	1,132	567
polish-companies-bankruptcy	binary	394	985	1,970	3,940	1,970
seismic-bumps	binary	172	431	861	1,722	862

I COMPLETE RESULTS, FAILURES, RUNTIME AND COST

This section lists every result of the benchmark: aggregate statistics of all pooled methods per group (§I.1), critical-difference diagrams (§I.2), the score of every method on every table (§I.3), cell coverage and every failed cell (§I.5), runtime (§I.6) and one-off cost (§I.7). The protocol, the statistics and the dated amendments are in Appendix A, the 71 tables in Tables 3 and 4, and the baseline configurations in Appendix B.

Which row is ours. TabEvo in the main text is the evolved program exactly as selected, fitted on a stratified subsample of at most 10,000 dev rows (its exact GPs are cubic in the rows; marked ‡ where subsampled). Amendment 7 designated a full-data variant as the representative in the full-data tables: the same program fitted on *all* dev rows, with the fitting changes of Amendment 7 (prediction pass-throughs reused; GP and metric hyperparameters searched on row subsamples; above 8,000 fit rows the final GP solve uses the subset-of-regressors / DTC mean on 8,000 inducing rows) and the stacking head of Amendments 13 and 13b (degenerate blocks excluded, inputs clamped to their fitted range, temperature softmax for class probabilities); it completed 10 of the 26 fresh tables, so it is reported here where it has a cell and nowhere in the main text. Other rows: the agent-designed seed, fitted as TabEvo; and the two pruned programs of Appendix F (speed A and B). In the released result records, TabEvo, the full-data variant and the seed are `ours_kvw_battery`, `ours_full` and `ours_seed_kvw`, and the suffixes `_s1` and `_s2` mark seed replicates. In pooled statistics each of the strongest trained configurations is represented by its five-fold bagged version (Amendment 5); the unbagged versions appear in the per-table results. Neural and kernel baselines receive z-scored numeric inputs (Amendment 4).

I.1 AGGREGATE RESULTS PER GROUP

Tables 44–48 give, for every method the aggregator pools, the Elo with its bootstrap interval, the normalised score, the average rank, the mean score, the number of failed cells and the median time, per group: fresh (the out-of-sample verdict), held-out, battery (in-sample for our architecture), all 51 TabArena tables, and the pre-registered high-dimensional group. Failed cells are scored as the table's worst method. Two robustness columns repeat the Elo on the complete-case tables (every method succeeded) and under TabArena's metrics (ROC-AUC for binary, log-loss for multiclass, RMSE for regression). Table 49 gives the secondary equal-data analysis, with every method limited to 4,000 dev rows, and Figure 23 compares each method's average rank in the two regimes on the same tables.

Table 44: All pooled methods on the fresh tables, full data, primary metric (failed cells scored as the table's worst method). Elo: Bradley–Terry on per-table pairwise outcomes, RF (D) = 1000, 95% interval from 1,000 bootstrap resamples of tables; Norm.: mean per-table min–max normalised score; Rank: average rank; Mean: mean score (×100); Fail: failed cells; Time: median fit+predict seconds per table. Elo_{cc}: Elo on the 26 tables where every method succeeded; Elo_{TA}: Elo under TabArena's metrics (ROC-AUC binary, log-loss multiclass, RMSE regression). n: tables with the method's results (of 26). Methods below the aggregator's coverage threshold (0.75 of the cells) are not pooled: Ensemble (TD), GP (ARD), xRFM (D), MLP (D), MLP-PLR (D), ResNet (D), SNN, DCNv2, NODE, TabNet, AutoInt, FT-Transformer (D), SAINT, ExcelFormer, TabR (D), RealTabR (D), ModernNCA, TabM (D), RealMLP (TD), Mitra, TabDPT, LimiX.

Method	Family	n	Elo [95% CI]	Norm.	Rank	Mean	Fail	Time (s)	Elo_{cc}	Elo_{TA}
TabFM	FM	26	1427 [1326, 1580]	0.94	3.5	90.2	0	74	1427	1783
TabPFN v3	FM	26	1334 [1267, 1438]	0.91	4.9	90.0	0	20	1334	1703
TabPFN v2.5	FM	26	1296 [1216, 1399]	0.88	5.5	89.9	0	15	1296	1556
TabICLv2	FM	26	1277 [1210, 1368]	0.88	5.8	89.9	0	26	1277	1587
TabICL v1.1	FM	26	1242 [1188, 1317]	0.87	6.5	89.8	0	17	1242	1460
CatBoost (TD, bag 5)	GBDT	26	1204 [1124, 1288]	0.83	7.2	89.7	0	24	1204	1393
TabPFN v2	FM	26	1165 [1069, 1263]	0.82	8.0	89.6	0	15	1165	1292
LightGBM (TD, bag 5)	GBDT	26	1164 [1092, 1238]	0.79	8.0	89.4	0	25	1164	1276
CatBoost (D)	GBDT	26	1134 [1074, 1203]	0.80	8.6	89.4	0	7.7	1134	1277
XGBoost (TD, bag 5)	GBDT	26	1035 [959, 1107]	0.70	10.7	89.1	0	24	1035	1215
HistGB (D)	GBDT	26	1016 [917, 1104]	0.66	11.1	89.2	0	1.3	1016	1107
LightGBM (D)	GBDT	26	1013 [927, 1090]	0.69	11.1	89.2	0	1.9	1013	1060
RF (D)	Classical	26	1000 [909, 1082]	0.65	11.4	89.0	0	1.3	1000	1000
XGBoost (D)	GBDT	26	992 [933, 1049]	0.69	11.5	89.1	0	1.6	992	1043
SVM	Classical	26	935 [818, 1026]	0.61	12.6	88.6	0	5.5	935	905
ExtraTrees (D)	Classical	26	903 [802, 988]	0.57	13.2	88.7	0	0.78	903	893
Linear	Classical	26	786 [659, 882]	0.38	15.0	87.5	0	0.20	786	911
k-NN	Classical	26	666 [492, 783]	0.25	16.3	86.8	0	0.22	666	414

Table 45: As Table 44, on the held-out tables (10; n = 1 per method; Elo_{cc} on 1 complete-case tables).

Method	Family	n	Elo [95% CI]	Norm.	Rank	Mean	Fail	Time (s)	Elo_{cc}	Elo_{TA}
TabFM	FM	1	3891 [3891, 3891]	1.00	1.0	89.9	0	65	3891	3891
TabPFN v3	FM	1	3508 [3508, 3508]	0.97	2.0	89.1	0	16	3508	3508
TabICLv2	FM	1	3207 [3207, 3207]	0.94	3.0	88.4	0	29	3207	3207
TabEvo (full data)	Ours	1	2944 [2944, 2944]	0.91	4.0	87.6	0	443	2944	2944
TabPFN v2.5	FM	1	2703 [2703, 2703]	0.90	5.0	87.5	0	16	2703	2703
CatBoost (TD, bag 5)	GBDT	1	2477 [2477, 2477]	0.84	6.0	85.8	0	19	2477	2477
LightGBM (TD, bag 5)	GBDT	1	2260 [2260, 2260]	0.83	7.0	85.6	0	15	2260	2260
TabPFN v2	FM	1	2049 [2049, 2049]	0.82	8.0	85.3	0	17	2049	2049
XGBoost (TD, bag 5)	GBDT	1	1842 [1842, 1842]	0.82	9.0	85.3	0	15	1842	1842
CatBoost (D)	GBDT	1	1635 [1635, 1635]	0.76	10.0	83.6	0	6.1	1635	1635
HistGB (D)	GBDT	1	1428 [1428, 1428]	0.75	11.0	83.4	0	1.1	1428	1428
LightGBM (D)	GBDT	1	1217 [1217, 1217]	0.75	12.0	83.4	0	1.4	1217	1217
RF (D)	Classical	1	1000 [1000, 1000]	0.74	13.0	83.3	0	1.8	1000	1000
XGBoost (D)	GBDT	1	774 [774, 774]	0.73	14.0	83.0	0	1.4	774	774
ExtraTrees (D)	Classical	1	533 [533, 533]	0.70	15.0	82.2	0	0.95	533	533
SVM	Classical	1	271 [271, 271]	0.53	16.0	78.0	0	5.6	271	271
k-NN	Classical	1	−31 [−31, −31]	0.38	17.0	74.1	0	0.26	−31	−31
Linear	Classical	1	−414 [−414, −414]	0.00	18.0	64.2	0	0.14	−414	−414

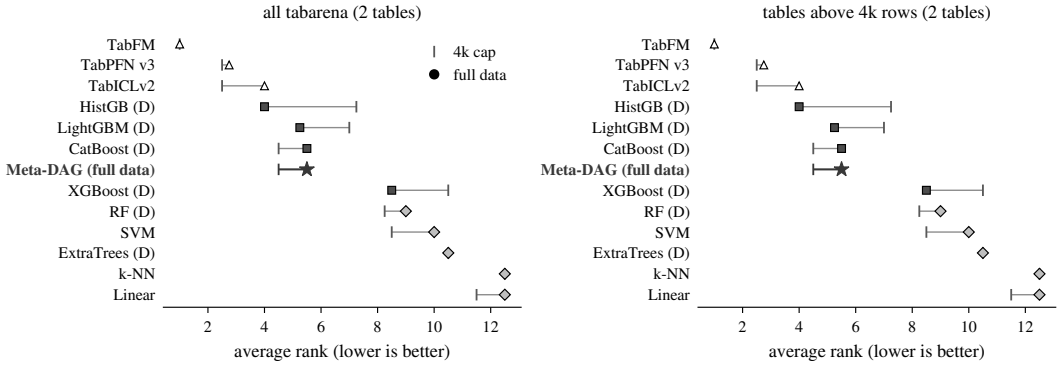


Figure 23: Average rank of every pooled method with every method limited to 4,000 dev rows (tick) and on full data (marker), over the same tables and methods in both regimes. Left: all TabArena tables; right: the tables above 4,000 dev rows, the only ones that differ between the regimes. TabEvo is labelled Meta-DAG.

Table 46: As Table 44, on the battery tables (15; n = 1 per method; Elo_{cc} on 1 complete-case tables). In-sample for our architecture.

Method	Family	n	Elo [95% CI]	Norm.	Rank	Mean	Fail	Time (s)	Elo_{cc}	Elo_{TA}
TabFM	FM	1	4260 [4260, 4260]	1.00	1.0	82.5	0	36	4260	4117
HistGB (D)	GBDT	1	3873 [3873, 3873]	0.96	2.0	82.3	0	1.4	3873	2276
TabPFN v2.5	FM	1	3570 [3570, 3570]	0.95	3.0	82.3	0	6.6	3570	3170
LightGBM (D)	GBDT	1	3209 [3209, 3209]	0.94	4.5	82.3	0	0.98	3209	1443
TabPFN v3	FM	1	3209 [3209, 3209]	0.94	4.5	82.3	0	14	3209	3433
LightGBM (TD, bag 5)	GBDT	1	2885 [2885, 2885]	0.93	6.0	82.3	0	16	2885	2930
TabICLv2	FM	1	2667 [2667, 2667]	0.93	7.0	82.3	0	11	2667	3734
CatBoost (D)	GBDT	1	2453 [2453, 2453]	0.92	8.0	82.2	0	2.7	2453	2068
XGBoost (TD, bag 5)	GBDT	1	2245 [2245, 2245]	0.91	9.0	82.2	0	8.3	2245	2487
TabEvo (full data)	Ours	1	1949 [1949, 1949]	0.91	10.5	82.2	0	905	1949	1862
TabPFN v2	FM	1	1949 [1949, 1949]	0.91	10.5	82.2	0	15	1949	2703
XGBoost (D)	GBDT	1	1560 [1560, 1560]	0.90	12.5	82.2	0	1.1	1560	1226
CatBoost (TD, bag 5)	GBDT	1	1560 [1560, 1560]	0.90	12.5	82.2	0	17	1560	1654
SVM	Classical	1	1242 [1242, 1242]	0.83	14.0	82.0	0	0.22	1242	196
RF (D)	Classical	1	1000 [1000, 1000]	0.79	15.0	81.8	0	0.76	1000	1000
ExtraTrees (D)	Classical	1	734 [734, 734]	0.60	16.0	81.2	0	1.4	734	760
Linear	Classical	1	431 [431, 431]	0.57	17.0	81.1	0	0.11	431	497
k-NN	Classical	1	44 [44, 44]	0.00	18.0	79.3	0	0.14	44	−188

Table 47: As Table 44, on the 51 TabArena tables (51; n = 2 per method; Elo_{cc} on 2 complete-case tables).

Method	Family	n	Elo [95% CI]	Norm.	Rank	Mean	Fail	Time (s)	Elo_{cc}	Elo_{TA}
TabFM	FM	2	2939 [2939, 4165]	1.00	1.0	86.2	0	66	2939	3708
TabPFN v3	FM	2	2215 [2215, 3705]	0.95	3.2	85.7	0	17	2215	3090
TabPFN v2.5	FM	2	2125 [2125, 3239]	0.92	4.0	84.9	0	14	2125	2534
TabICLv2	FM	2	2027 [2027, 3336]	0.93	5.0	85.3	0	20	2027	3090
LightGBM (TD, bag 5)	GBDT	2	1902 [1902, 2413]	0.88	6.5	84.0	0	18	1902	2298
HistGB (D)	GBDT	2	1902 [1126, 3612]	0.85	6.5	82.9	0	1.3	1902	1900
TabEvo (full data)	Ours	2	1845 [1250, 3012]	0.91	7.2	84.9	0	674	1845	2180
LightGBM (D)	GBDT	2	1768 [863, 2804]	0.84	8.2	82.9	0	1.5	1768	1408
XGBoost (TD, bag 5)	GBDT	2	1709 [1611, 1709]	0.87	9.0	83.7	0	16	1709	2071
CatBoost (D)	GBDT	2	1709 [1384, 1873]	0.84	9.0	82.9	0	4.4	1709	1900
CatBoost (TD, bag 5)	GBDT	2	1688 [786, 2433]	0.87	9.2	84.0	0	21	1688	2017
TabPFN v2	FM	2	1688 [1250, 1901]	0.86	9.2	83.7	0	15	1688	2180
XGBoost (D)	GBDT	2	1149 [312, 1149]	0.81	13.2	82.6	0	1.4	1149	1000
RF (D)	Classical	2	1000 [98, 1000]	0.77	14.0	82.6	0	1.2	1000	1000
SVM	Classical	2	799 [−310, 799]	0.68	15.0	80.0	0	3.2	799	115
ExtraTrees (D)	Classical	2	688 [−231, 688]	0.65	15.5	81.7	0	0.89	688	564
k-NN	Classical	2	41 [−1068, 41]	0.19	17.5	76.7	0	0.19	41	−125
Linear	Classical	2	41 [−1138, 41]	0.28	17.5	72.7	0	0.14	41	−3

Table 48: As Table 44, on the high-dim. tables (20; n = 16 per method; Elo_{cc} on 16 complete-case tables). Pre-registered hypothesis H1 (Amendment 2).

Method	Family	n	Elo [95% CI]	Norm.	Rank	Mean	Fail	Time (s)	Elo_{cc}	Elo_{TA}
TabFM	FM	16	1448 [1286, 1809]	0.94	2.5	83.4	0	132	1448	1637
TabPFN v3	FM	16	1317 [1178, 1540]	0.84	3.9	82.5	0	24	1317	1360
TabICLv2	FM	16	1209 [1103, 1368]	0.82	5.5	81.4	0	37	1209	1174
CatBoost (TD, bag 5)	GBDT	16	1198 [1105, 1299]	0.79	5.7	81.1	0	605	1198	1165
XGBoost (TD, bag 5)	GBDT	16	1169 [1080, 1255]	0.80	6.1	80.9	0	54	1169	1130
LightGBM (TD, bag 5)	GBDT	16	1109 [994, 1215]	0.67	7.2	78.7	0	24	1109	1096
HistGB (D)	GBDT	16	1070 [991, 1129]	0.71	7.8	79.0	0	4.0	1070	1038
CatBoost (D)	GBDT	16	1009 [924, 1085]	0.69	8.9	79.3	0	19	1009	968
RF (D)	Classical	16	1000 [924, 1053]	0.69	9.1	79.0	0	1.5	1000	1000
Linear	Classical	16	973 [798, 1114]	0.51	9.6	74.2	0	0.22	973	937
ExtraTrees (D)	Classical	16	963 [822, 1080]	0.55	9.8	75.0	0	0.87	963	891
LightGBM (D)	GBDT	16	944 [803, 1055]	0.58	10.1	77.6	0	2.2	944	860
XGBoost (D)	GBDT	16	920 [804, 1007]	0.57	10.5	78.1	0	3.5	920	850
SVM	Classical	16	920 [807, 1022]	0.55	10.5	76.0	0	7.2	920	881
k-NN	Classical	16	743 [484, 895]	0.22	12.9	70.2	0	0.25	743	443

Table 49: The secondary equal-data analysis: every method fitted on at most 4,000 dev rows (the 26 larger tables subsampled, stratified for classification, seed 0; test splits unchanged), primary metric, failed cells scored as worst. Columns as in Table 44. Only the methods the aggregator pools in this regime are listed.

Method	Family	n	Elo [95% CI]	Norm.	Rank	Mean	Fail	Time (s)
Fresh (26 tables)								
TabFM	FM	25	1271 [1193, 1388]	0.82	2.0	89.6	0	44
TabPFN v3	FM	25	1153 [1081, 1234]	0.59	2.8	89.3	0	9.2
TabICLv2	FM	25	1132 [1065, 1212]	0.60	2.9	89.3	0	8.5
TabEvo evolved ($\leq$4k rows)	Ours	25	1032 [933, 1103]	0.32	3.6	89.0	1	465
TabEvo seed ($\leq$4k rows)	Ours	25	1000 [903, 1066]	0.27	3.8	88.9	1	344
All TabArena (51 tables)								
TabFM	FM	50	1372 [1302, 1465]	0.84	1.8	86.2	0	41
TabPFN v3	FM	50	1230 [1181, 1285]	0.64	2.6	85.8	0	9.4
TabICLv2	FM	50	1198 [1145, 1255]	0.60	2.8	85.5	0	8.3
TabEvo evolved ($\leq$4k rows)	Ours	50	1068 [1001, 1119]	0.28	3.6	85.2	1	362
TabEvo seed ($\leq$4k rows)	Ours	50	1000 [920, 1062]	0.19	4.0	85.0	1	319

I.2 CRITICAL-DIFFERENCE DIAGRAMS

Figures 24 and 25 show the Friedman test with the Nemenyi critical difference (CD, $\alpha = 0.05$) over all pooled methods of each group. Following Benavoli et al. (2016), these mean-rank diagrams are descriptive; claims of superiority or inferiority rest on the Holm-corrected pairwise Wilcoxon tests of the main text.

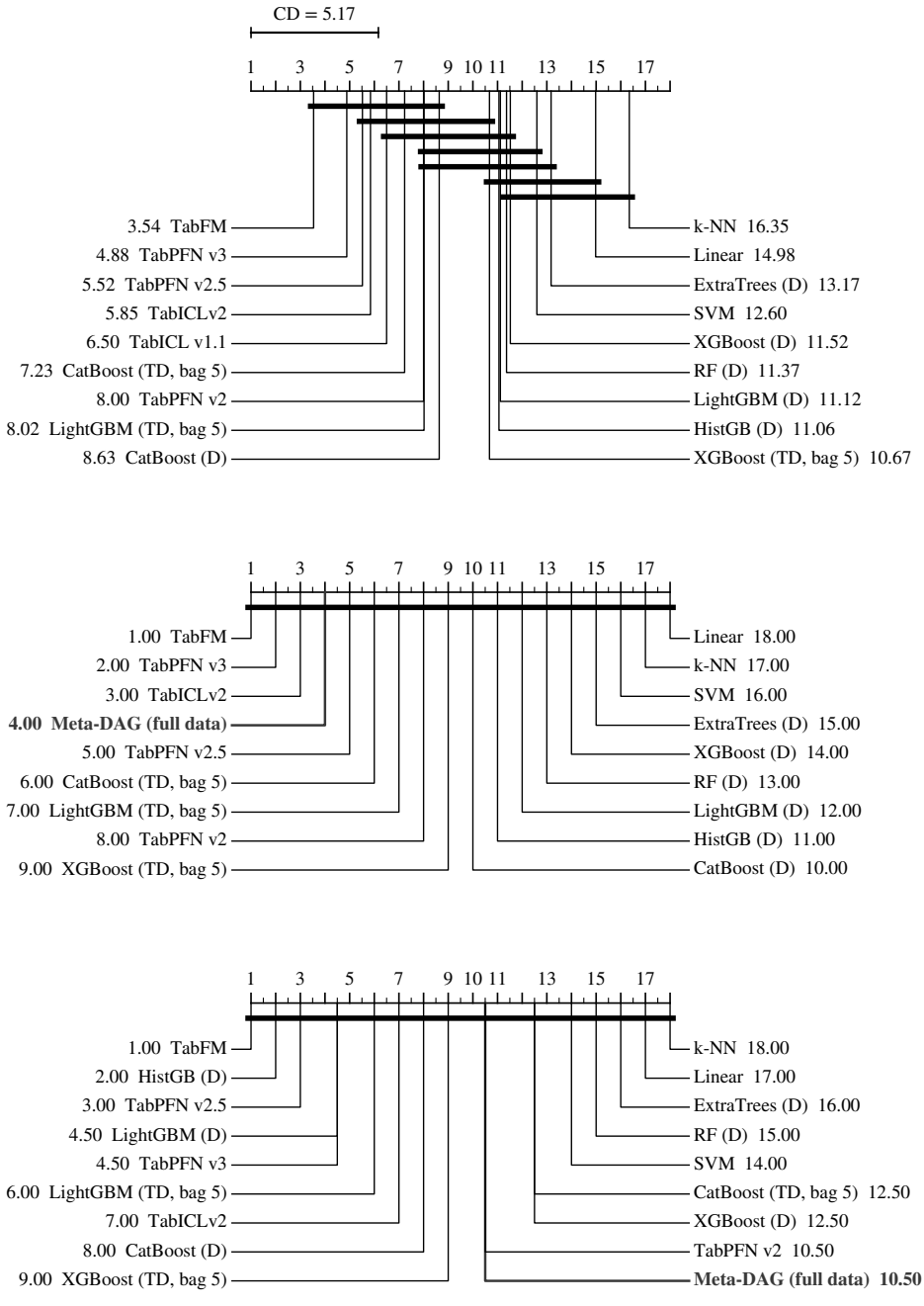


Figure 24: Critical-difference diagrams, full data, primary metric, failed cells scored as worst: fresh (top), held-out (middle) and battery tables (bottom; in-sample for our architecture). A bar joins methods whose average ranks differ by less than the CD. TabEvo is labelled Meta-DAG.

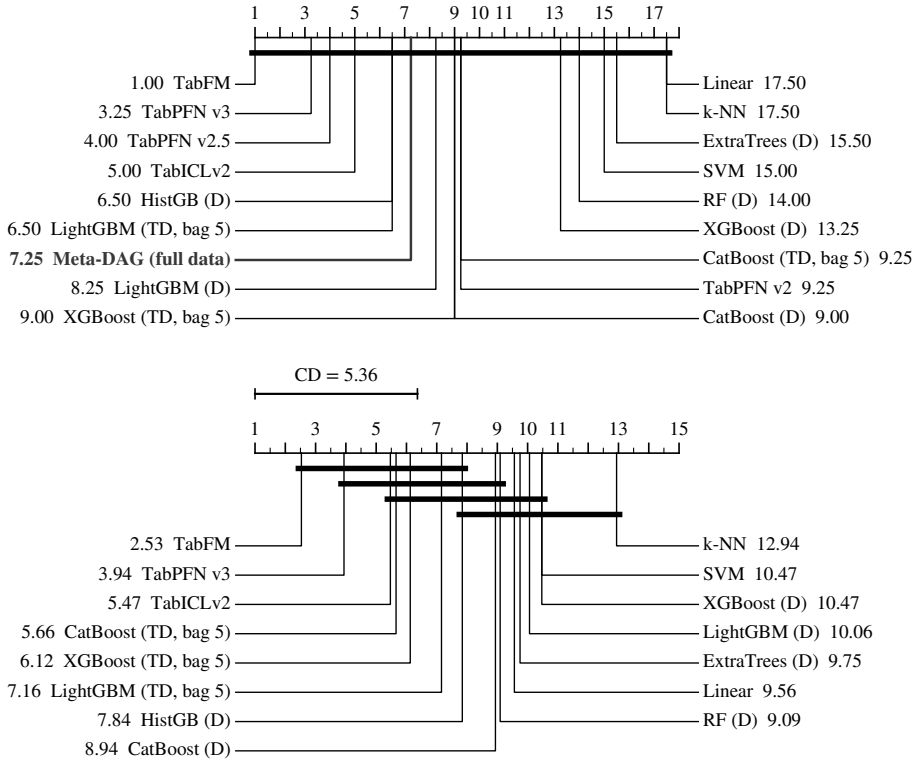


Figure 25: Critical-difference diagrams as in Figure 24: all 51 TabArena tables (top) and the pre-registered high-dimensional tables (bottom).

Table 50: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 1 of 4 (Ours, FM), fresh (26) and held-out (10) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 325 ok, 3 failed, 5 n/a, 171 missing.

Table		Ours (full data)	Ours (evolved, 10k)	Ours seed (10k)	Ours speed A	Ours speed B	TabICLv1.1	TabICLv2	TabPFNv2	TabPFNv2.5	TabPFNv3	TabFM	Mitra	TabDPT	LimiX
Fresh (26)															
amazon-employee-access	Acc	–	–	×	–	–	94.8	95.0	94.2	94.8	94.9	94.8	–	–	–
anneal	Acc	–	99.0	99.0	99.3	99.3	99.3	99.3	**99.7**	**99.7**	**99.7**	**99.7**	99.3	98.0	**99.7**
apsfailure	Acc	–	99.1‡	99.1‡	99.1‡	99.1‡	99.5	99.5	99.2	99.4	99.4	**99.5**	–	–	–
bank-customer-churn	Acc	–	86.8	86.9	86.9	86.7	86.5	86.7	86.9	**87.0**	86.9	86.7	–	–	–
bioresponse	Acc	–	81.0	80.1	80.4	80.4	79.5	79.5	78.3	80.3	80.7	**81.1**	–	–	–
churn	Acc	–	96.6	96.1	96.0	95.7	95.9	97.2	97.1	97.2	97.6	**98.2**	–	–	–
coil2000-insurance-poli.	Acc	–	94.0	94.0	94.0	94.0	94.0	94.1	94.0	94.1	94.0	94.1	–	–	–
credit-g	Acc	–	76.3	76.9	72.8	72.8	75.1	75.1	75.7	74.9	75.4	77.2	–	–	–
customer-satisfaction-i.	Acc	–	94.5‡	94.5‡	94.2‡	94.2‡	96.0	96.3	95.0	96.0	96.4	**96.7**	–	–	–
diabetes130us	Acc	–	–	–	91.2‡	91.2‡	91.2	91.2	91.2	91.2	91.2	**91.3**	–	–	–
e-commereshippingdata	Acc	–	67.8	67.8	67.3	67.5	68.5	68.6	68.9	68.8	68.9	**69.0**	–	–	–
fitness-club	Acc	–	76.4	76.4	76.2	76.4	**78.0**	77.6	**78.0**	**78.0**	77.8	77.6	–	–	–
givemesomecredit	Acc	–	93.6‡	–	93.6‡	93.6‡	93.7	93.7	93.7	93.8	93.8	**93.9**	–	–	–
hiva-agnostic	Acc	–	×	×	**96.5**	**96.5**	**96.5**	**96.5**	**96.5**	**96.5**	96.4	**96.5**	–	–	–
hr-analytics-job-change.	Acc	–	79.4‡	79.3‡	79.2‡	79.0‡	80.4	80.5	80.3	80.3	80.7	**80.8**	–	–	–
in-vehicle-coupon-recom.	Acc	–	78.2	78.0	78.3	77.5	77.8	77.7	75.3	**78.3**	78.1	77.8	–	–	–
is-this-a-good-customer	Acc	–	88.5	88.3	**88.7**	88.5	**88.7**	88.0	88.5	88.5	88.3	87.7	–	–	–
jm1	Acc	–	81.7	81.8	81.5	81.4	82.4	82.6	81.3	82.0	82.6	**82.7**	–	–	–
kddcup09-appetency	Acc	–	–	–	–	–	**98.2**	**98.2**	**98.2**	**98.2**	**98.2**	**98.2**	–	–	–
marketing-campaign	Acc	–	89.0	89.0	89.4	89.4	88.9	90.0	**90.4**	89.7	89.3	90.2	–	–	–
mic	Acc	–	88.7	88.9	88.5	88.5	88.7	89.1	89.1	88.7	88.5	**89.2**	–	–	–
naticusdroid	Acc	–	94.5	94.5	94.2	94.2	**94.6**	94.5	94.6	94.5	94.4	94.6	–	–	–
polish-companies-bankru.	Acc	–	96.2	96.0	95.4	95.4	97.2	98.0	96.6	98.2	98.0	**99.2**	–	–	–
sdss17	Acc	–	–	–	–	–	97.6	97.8	97.5	97.7	**98.0**	97.9	–	–	–
seismic-bumps	Acc	–	**93.4**	**93.4**	**93.4**	**93.4**	**93.4**	**93.4**	**93.4**	**93.4**	**93.4**	**93.4**	–	–	–
taiwanese-bankruptcy-pr.	Acc	–	97.1	97.1	96.9	96.9	97.2	97.2	97.0	97.1	97.3	97.1	–	–	–
Held-out (10)															
another-dataset-on-used.	R^2	–	86.7	86.6	–	–	n/a	87.0	86.8	87.1	87.3	**87.9**	–	–	–
bank-marketing	Acc	–	89.4‡	89.4‡	–	–	89.4	89.4	89.4	89.4	89.4	89.3	–	–	–
diabetes	Acc	–	77.7	78.5	–	–	75.8	74.6	75.4	75.8	75.4	75.8	75.4	77.0	**78.9**
diamonds	R^2	–	98.0‡	97.9‡	–	–	n/a	98.5	98.3	98.5	98.5	**98.5**	–	–	–
food-delivery-time	R^2	–	32.8‡	24.8‡	–	–	n/a	33.7	34.6	35.1	36.0	**39.1**	–	–	–
hazelnut-spread-contami.	Acc	–	94.0	94.0	–	–	95.9	96.4	94.6	95.8	96.8	**98.0**	–	–	–
houses	R^2	87.6	87.6‡	87.6‡	–	–	n/a	88.4	85.3	87.5	89.1	**89.9**	–	–	–
online-shoppers-intenti.	Acc	–	89.6	89.5	–	–	90.0	90.1	89.9	89.9	90.2	**90.4**	–	–	–
qsar-tid-11	R^2	–	72.7	71.7	–	–	n/a	69.9	68.2	72.4	74.2	**74.6**	–	–	–
splice	Acc	–	96.4	96.4	–	–	96.2	97.8	**97.9**	97.5	97.8	97.8	–	–	–

I.3 PER-TABLE RESULTS

Tables 50–57 give the test score of every method on every table (primary metric, full data, one fit with seed 0). Methods are split into column blocks by family; within each block, the rows are the fresh and held-out tables (first table) and the battery and high-dimensional tables (second table). The bold entry is the best of all methods on that table, across blocks. Ablations are in Appendix E; the seed replicates used for the noise floor are not shown.

Table 51: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 1 of 4 (Ours, FM), battery (15) and high-dim. (20) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 192 ok, 2 failed, 8 n/a, 288 missing.

Table		Ours (full data)	Ours (evolved, 10k)	Ours seed (10k)	Ours speed A	Ours speed B	TabICLv1.1	TabICLv2	TabPFNv2	TabPFNv2.5	TabPFNv3	TabFM	Mitra	TabDPT	LimiX
Battery (15)															
airfoil-self-noise	R^2	–	97.7	97.7	–	–	n/a	97.9	98.1	98.0	**98.6**	98.2	–	–	–
blood-transfusion-servi.	Acc	–	80.0	78.4	–	–	78.8	79.2	79.2	78.8	78.0	80.0	78.8	80.0	79.2
concrete-compressive-st.	R^2	–	93.6	92.8	–	–	n/a	95.3	94.4	94.8	**95.5**	94.8	–	–	–
credit-card-clients-def.	Acc	82.2	82.1‡	82.0‡	–	–	82.2	82.3	82.2	82.3	82.3	**82.5**	–	–	–
healthcare-insurance-ex.	R^2	–	87.0	86.9	–	–	n/a	87.7	85.6	87.4	87.6	**87.9**	–	–	–
heloc	Acc	–	73.6	73.6	–	–	73.0	73.1	73.6	73.5	**73.7**	73.2	–	–	–
maternal-health-risk	Acc	–	86.7	86.7	–	–	**88.8**	**88.8**	86.4	85.8	87.9	88.5	–	–	–
miami-housing	R^2	–	93.0	92.6	–	–	n/a	93.9	92.9	93.7	94.0	**94.5**	–	–	–
physiochemical-protein	R^2	–	68.7‡	68.6‡	–	–	n/a	75.5	68.9	75.2	**76.9**	76.7	–	–	–
qsar-biodeg	Acc	–	88.9	88.6	–	–	88.4	**89.5**	87.8	88.4	88.9	**89.5**	–	–	–
qsar-fish-toxicity	R^2	–	63.0	63.1	–	–	n/a	64.5	64.0	64.3	64.6	65.8	64.9	63.1	**66.1**
students-dropout-and-ac.	Acc	–	77.8	77.4	–	–	76.6	77.8	78.6	78.8	77.4	**79.1**	–	–	–
superconductivity	R^2	–	91.7‡	91.6‡	–	–	n/a	93.1	92.0	92.7	93.0	**93.2**	–	–	–
website-phishing	Acc	–	91.1	91.1	–	–	92.9	93.3	92.7	92.7	92.9	**93.6**	–	–	–
wine-quality	R^2	–	53.4	53.4	–	–	n/a	52.8	35.0	47.3	53.6	**54.2**	–	–	–
High-dim. (20)															
ap-breast-ovary	Acc	–	–	–	–	–	–	94.5	–	–	96.3	95.4	–	–	–
ap-colon-kidney	Acc	–	–	–	–	–	–	98.2	–	–	98.2	98.2	–	–	–
arcene	Acc	–	85.0	85.0	–	–	–	80.0	–	–	82.5	82.5	–	–	–
christine	Acc	–	–	75.1	–	–	–	76.8	–	–	77.5	**78.0**	–	–	–
cnae-9	Acc	–	94.4	95.4	–	–	–	95.8	–	–	95.8	**98.6**	–	–	–
communities-and-crime	R^2	–	67.3	66.9	–	–	–	68.9	–	–	**69.2**	69.1	–	–	–
dilbert	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–
gina-agnostic	Acc	–	97.1	97.1	–	–	–	96.8	–	–	97.0	**99.0**	–	–	–
gisette	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–
internet-advertisements	Acc	–	$\times$	$\times$	–	–	–	97.0	–	–	95.9	**97.1**	–	–	–
madelon	Acc	–	92.1	92.7	–	–	–	**93.3**	–	–	89.6	**93.3**	–	–	–
mfeat-factors	Acc	–	**99.2**	**99.2**	–	–	–	99.0	–	–	99.0	**99.2**	–	–	–
ova-lung	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–
qsar-tid-10980	R^2	–	68.5	67.2	–	–	–	62.9	–	–	69.9	**70.3**	–	–	–
qsar-tid-194	R^2	–	79.1	77.8	–	–	–	76.7	–	–	78.9	**80.2**	–	–	–
qsar-tid-72	R^2	–	63.2	61.5	–	–	–	56.5	–	–	62.9	**63.9**	–	–	–
santander-transaction-v.	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–
semeion	Acc	–	96.6	97.2	–	–	–	97.2	–	–	97.8	**98.4**	–	–	–
tecator	R^2	–	99.9	99.8	–	–	–	99.9	–	–	**99.9**	99.9	–	–	–
topo-2-1	R^2	–	6.0	6.1	–	–	–	8.5	–	–	9.3	**11.0**	–	–	–

Table 52: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 2 of 4 (GBDT), fresh (26) and held-out (10) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 376 ok, 0 failed, 0 n/a, 20 missing.

Table		XGB	XGB-TD	LGBM	LGBM-TD	CB	CB-TD	CB-TD-b5	LGBM-TD-b5	XGB-TD-b5	HGB	Ens-TD
Fresh (26)												
amazon-employee-access	Acc	94.5	94.4	94.8	94.7	94.9	95.0	**95.3**	95.0	94.6	94.4	–
anneal	Acc	99.0	99.3	99.3	97.7	99.0	**99.7**	**99.7**	98.0	99.3	98.7	97.7
apsfailure	Acc	99.4	99.4	99.2	99.4	99.3	99.3	99.3	99.4	99.5	99.3	–
bank-customer-churn	Acc	86.6	86.5	86.7	86.8	86.7	86.7	86.6	86.7	86.4	86.2	–
bioresponse	Acc	78.6	79.8	80.7	79.5	80.7	79.7	81.0	80.7	79.8	79.5	79.9
churn	Acc	95.7	94.4	96.0	94.4	95.5	95.4	95.5	95.2	94.8	95.7	95.4
coil2000-insurance-poli.	Acc	94.0	94.0	93.6	93.3	**94.1**	93.9	94.0	94.0	94.0	92.9	–
credit-g	Acc	74.9	**77.8**	74.0	76.6	74.6	76.0	76.0	75.1	75.7	77.2	**77.8**
customer-satisfaction-i.	Acc	95.6	95.7	95.3	95.7	95.7	95.8	95.9	95.7	95.8	95.3	–
diabetes130us	Acc	91.2	91.2	91.2	91.2	91.2	91.2	91.2	91.2	91.2	91.2	–
e-commereshippingdata	Acc	67.8	66.7	66.2	66.8	68.0	68.3	68.0	66.8	66.8	65.9	–
fitness-club	Acc	75.8	76.4	76.8	76.6	77.0	76.8	77.8	76.6	75.2	75.8	77.6
givemesomecredit	Acc	93.7	93.8	93.8	93.8	93.8	93.8	93.8	93.8	93.8	93.8	–
hiva-agnostic	Acc	96.3	96.3	96.3	**96.5**	**96.5**	96.3	**96.5**	**96.5**	96.3	96.1	**96.5**
hr-analytics-job-change.	Acc	79.2	78.7	78.8	79.1	80.1	79.6	80.0	79.1	79.2	79.8	–
in-vehicle-coupon-recom.	Acc	75.4	76.3	75.0	76.1	76.0	76.8	77.7	76.3	77.6	75.8	–
is-this-a-good-customer	Acc	87.1	88.5	86.8	88.3	88.0	87.5	87.7	88.5	87.3	87.0	88.3
jm1	Acc	80.7	81.3	81.4	81.2	81.6	81.8	81.3	81.6	81.5	81.7	–
kddcup09-appetency	Acc	**98.2**	**98.2**	98.2	98.2	98.2	**98.2**	**98.2**	**98.2**	**98.2**	98.1	–
marketing-campaign	Acc	88.2	85.5	89.3	90.1	88.9	88.8	88.5	89.6	85.5	89.0	88.0
mic	Acc	87.8	88.2	88.2	87.5	87.8	88.4	88.4	87.7	88.4	88.5	88.4
naticusdroid	Acc	93.7	94.0	93.6	93.5	94.0	94.0	94.2	93.6	94.2	93.9	–
polish-companies-bankru.	Acc	96.5	95.5	96.0	95.6	95.9	95.7	96.1	95.8	95.4	97.0	95.8
sdss17	Acc	97.5	96.5	97.6	97.6	97.5	97.7	97.8	97.7	96.6	97.8	–
seismic-bumps	Acc	**93.4**	**93.4**	93.0	**93.4**	**93.4**	93.2	93.0	**93.4**	92.8	92.3	**93.4**
taiwanese-bankruptcy-pr.	Acc	97.0	97.3	97.1	96.8	97.2	97.2	97.3	**97.4**	97.2	97.2	–
Held-out (10)												
another-dataset-on-used.	R^2	83.5	84.5	84.7	84.1	85.8	85.2	86.6	85.5	85.4	84.3	86.3
bank-marketing	Acc	89.3	89.3	89.1	89.2	89.4	89.3	**89.4**	89.3	89.4	89.3	–
diabetes	Acc	76.6	76.6	77.3	76.2	76.6	77.7	77.0	75.8	76.6	73.8	71.1
diamonds	R^2	98.2	98.3	98.1	98.0	98.2	98.1	98.2	98.1	98.3	98.1	–
food-delivery-time	R^2	22.1	22.3	32.5	32.9	36.3	36.1	36.6	33.8	23.6	36.9	–
hazelnut-spread-contami.	Acc	89.8	91.1	89.9	89.8	89.2	90.2	91.4	91.1	92.0	91.4	93.0
houses	R^2	83.0	83.9	83.4	84.3	83.6	84.8	85.8	85.6	85.3	83.4	–
online-shoppers-intenti.	Acc	89.6	89.2	89.3	89.7	89.5	89.7	89.8	89.8	89.9	90.0	–
qsar-tid-11	R^2	67.4	69.8	67.9	70.7	66.5	68.9	71.0	72.4	72.2	69.5	71.2
splice	Acc	97.3	97.3	97.1	97.2	96.5	96.7	96.7	97.3	97.5	97.3	96.7

Table 53: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 2 of 4 (GBDT), battery (15) and high-dim. (20) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 359 ok, 0 failed, 0 n/a, 26 missing.

Table		XGB	XGB-TD	LGBM	LGBM-TD	CB	CB-TD	CB-TD-b5	LGBM-TD-b5	XGB-TD-b5	HGB	Ens-TD
Battery (15)												
airfoil-self-noise	R^2	92.4	93.8	92.1	95.3	92.3	94.5	95.1	95.9	94.9	93.9	96.7
blood-transfusion-servi.	Acc	77.2	75.2	77.6	77.6	77.6	78.0	77.2	76.4	79.2	77.2	80.0
concrete-compressive-st.	R^2	89.7	90.6	91.0	90.0	92.2	88.1	89.2	92.0	91.8	92.6	91.4
credit-card-clients-def.	Acc	82.2	81.9	82.3	82.5	82.2	82.1	82.2	82.3	82.2	82.3	–
healthcare-insurance-ex.	R^2	84.7	83.7	86.1	83.6	87.1	84.1	85.0	84.4	85.7	84.2	86.7
heloc	Acc	71.7	71.6	71.7	72.1	72.6	72.5	72.9	72.7	73.2	71.7	–
maternal-health-risk	Acc	81.7	81.7	81.1	72.5	81.1	80.8	86.1	80.8	86.4	85.8	74.3
miami-housing	R^2	90.2	91.0	90.7	91.0	92.4	91.1	92.5	92.1	91.9	91.6	–
physiochemical-protein	R^2	58.2	65.2	54.7	65.4	54.8	64.9	66.5	67.2	67.5	54.4	–
qsar-biodeg	Acc	85.2	86.4	85.8	80.7	86.1	83.8	86.4	86.6	88.4	88.1	86.4
qsar-fish-toxicity	R^2	51.3	57.6	58.8	56.4	60.1	57.8	60.9	61.9	60.9	59.0	60.7
students-dropout-and-ac.	Acc	76.0	76.1	76.3	76.5	76.3	76.5	76.9	77.6	76.8	76.9	77.3
superconductivity	R^2	90.9	91.9	90.6	91.7	90.1	91.7	92.4	92.4	92.4	90.5	–
website-phishing	Acc	89.1	89.1	89.8	89.8	87.6	89.6	90.7	89.1	91.8	90.0	89.4
wine-quality	R^2	40.4	47.2	42.5	48.5	42.9	47.4	51.0	50.7	50.4	43.2	–
High-dim. (20)												
ap-breast-ovary	Acc	97.2	**99.1**	98.2	97.2	97.2	**99.1**	98.2	98.2	98.2	97.2	–
ap-colon-kidney	Acc	96.4	97.3	95.5	96.4	96.4	97.3	98.2	96.4	98.2	98.2	–
arcene	Acc	77.5	80.0	82.5	80.0	82.5	82.5	85.0	72.5	85.0	80.0	–
christine	Acc	74.0	74.5	74.5	73.8	74.2	75.6	74.7	75.2	74.1	75.0	–
cnae-9	Acc	92.6	91.2	76.9	69.4	90.3	92.1	94.4	69.9	93.5	83.3	–
communities-and-crime	R^2	57.9	63.3	63.6	63.3	66.9	59.9	64.9	65.5	64.5	64.8	–
dilbert	Acc	98.0	99.0	98.6	99.1	97.7	99.1	**99.2**	99.0	**99.2**	98.9	–
gina-agnostic	Acc	95.5	94.4	94.5	95.5	95.2	95.4	96.3	95.8	94.8	95.5	–
gisette	Acc	97.5	97.7	97.6	97.8	97.9	97.2	**98.2**	98.1	98.0	98.1	–
internet-advertisements	Acc	96.8	96.6	97.0	96.8	96.8	96.8	96.5	96.8	97.0	97.0	–
madelon	Acc	79.2	76.3	79.2	74.4	83.7	85.4	85.8	79.8	80.0	80.6	–
mfeat-factors	Acc	95.8	97.2	96.0	98.0	97.8	97.8	98.2	97.8	97.8	97.8	–
ova-lung	Acc	98.7	**99.0**	**99.0**	98.7	**99.0**	98.4	**99.0**	**99.0**	98.7	98.7	–
qsar-tid-10980	R^2	61.9	64.8	61.8	64.5	61.0	64.6	67.5	67.8	67.7	63.0	–
qsar-tid-194	R^2	73.5	76.1	73.7	76.9	72.7	76.3	77.9	78.8	78.9	75.3	–
qsar-tid-72	R^2	56.3	60.2	57.9	60.2	57.8	60.5	63.3	62.6	62.3	59.4	–
santander-transaction-v.	R^2	18.8	19.9	23.1	20.8	21.3	19.2	24.0	**25.5**	22.4	22.3	–
semeion	Acc	91.8	95.6	92.5	92.5	94.0	94.0	95.3	95.6	96.2	93.7	–
tecator	R^2	99.2	98.9	95.5	99.2	96.9	94.1	95.0	99.4	99.1	97.8	–
topo-2-1	R^2	3.5	5.5	3.2	4.9	5.7	3.1	7.0	6.9	6.6	5.1	–

Table 54: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 3 of 4 (Classical), fresh (26) and held-out (10) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 188 ok, 0 failed, 0 n/a, 64 missing.

Table		RF	ET	kNN	Lin	SVM	GP	xRFM
Fresh (26)								
amazon-employee-access	Acc	94.6	94.0	94.5	94.5	94.5	93.2‡	94.7
anneal	Acc	99.3	98.3	96.0	97.7	97.7	–	–
apsfailure	Acc	99.4	99.4	98.9	99.1	98.3	–	99.3
bank-customer-churn	Acc	86.7	86.1	83.9	80.7	86.0	–	–
bioresponse	Acc	79.2	80.3	75.1	73.9	78.4	–	–
churn	Acc	95.5	93.5	89.1	87.1	92.3	–	–
coil2000-insurance-poli.	Acc	92.8	92.3	93.5	93.8	94.0	–	–
credit-g	Acc	76.6	73.7	72.8	74.6	72.2	–	–
customer-satisfaction-i.	Acc	94.9	95.5	93.1	90.3	93.9	94.6‡	–
diabetes130us	Acc	91.2	91.2	90.8	91.1	91.2	–	–
e-commereshippingdata	Acc	66.4	66.6	64.7	63.7	64.7	–	–
fitness-club	Acc	75.0	74.0	72.4	76.4	77.6	–	–
givemesomecredit	Acc	93.6	93.4	93.3	93.4	93.7	93.3‡	–
hiva-agnostic	Acc	96.4	96.2	**96.5**	95.2	**96.5**	–	–
hr-analytics-job-change.	Acc	78.2	77.6	76.6	78.4	78.9	–	–
in-vehicle-coupon-recom.	Acc	75.3	73.8	65.5	69.0	74.9	–	–
is-this-a-good-customer	Acc	87.8	88.0	87.0	88.2	88.5	–	–
jm1	Acc	81.7	81.6	80.0	81.4	81.3	–	–
kddcup09-appetency	Acc	**98.2**	**98.2**	98.2	98.0	**98.2**	–	–
marketing-campaign	Acc	88.4	88.0	87.0	89.3	88.8	–	–
mic	Acc	87.5	88.9	86.1	86.6	88.9	–	–
naticusdroid	Acc	93.3	92.7	93.0	93.4	94.2	–	–
polish-companies-bankru.	Acc	93.7	96.0	93.1	92.9	93.2	–	–
sdss17	Acc	97.7	97.2	85.9	95.4	94.6	–	–
seismic-bumps	Acc	92.9	93.0	93.2	93.3	**93.4**	–	–
taiwanese-bankruptcy-pr.	Acc	97.3	97.2	96.9	97.2	96.8	–	–
Held-out (10)								
another-dataset-on-used.	R^2	85.6	85.4	84.4	85.4	84.8	–	–
bank-marketing	Acc	88.7	87.5	88.1	89.3	89.3	89.2‡	88.7
diabetes	Acc	76.6	78.1	77.0	78.1	78.5	–	–
diamonds	R^2	98.0	98.2	96.1	92.0	97.0	–	–
food-delivery-time	R^2	29.8	15.4	13.2	16.9	23.1	–	33.6
hazelnut-spread-contami.	Acc	88.5	89.2	84.4	87.8	89.4	–	–
houses	R^2	83.3	82.2	74.1	64.2	78.0	–	–
online-shoppers-intenti.	Acc	89.6	89.4	87.0	87.9	88.9	–	–
qsar-tid-11	R^2	70.7	50.0	59.9	55.4	67.8	–	–
splice	Acc	95.3	95.9	78.3	94.8	96.9	–	–

Table 55: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 3 of 4 (Classical), battery (15) and high-dim. (20) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 176 ok, 0 failed, 0 n/a, 69 missing.

Table		RF	ET	kNN	Lin	SVM	GP	xRFM
Battery (15)								
airfoil-self-noise	R^2	93.1	95.3	70.6	53.4	77.7	–	–
blood-transfusion-servi.	Acc	76.4	75.2	75.2	77.6	78.0	–	–
concrete-compressive-st.	R^2	89.3	91.2	70.8	64.7	82.2	–	–
credit-card-clients-def.	Acc	81.8	81.2	79.3	81.1	82.0	82.2‡	–
healthcare-insurance-ex.	R^2	84.3	84.6	76.7	76.6	86.3	–	–
heloc	Acc	72.1	72.3	67.6	72.1	72.8	–	–
maternal-health-risk	Acc	85.8	85.5	66.9	61.5	69.5	–	–
miami-housing	R^2	90.2	91.7	88.1	70.6	90.2	–	–
physiochemical-protein	R^2	65.1	67.4	58.1	29.1	40.8	–	–
qsar-biodeg	Acc	87.5	87.2	85.2	86.1	86.9	–	–
qsar-fish-toxicity	R^2	60.5	62.1	60.9	57.0	60.8	–	–
students-dropout-and-ac.	Acc	75.4	76.3	69.2	76.4	75.9	–	–
superconductivity	R^2	92.0	92.2	89.4	73.1	83.1	–	–
website-phishing	Acc	90.2	88.0	87.4	85.8	91.1	–	–
wine-quality	R^2	48.5	51.1	32.8	28.5	38.5	–	–
High-dim. (20)								
ap-breast-ovary	Acc	96.3	95.4	91.7	96.3	95.4	–	–
ap-colon-kidney	Acc	97.3	98.2	**99.1**	98.2	97.3	–	–
arcene	Acc	82.5	85.0	75.0	**87.5**	75.0	–	–
christine	Acc	73.2	73.4	67.4	67.7	74.9	–	–
cnae-9	Acc	90.3	91.7	69.9	94.9	75.9	–	–
communities-and-crime	R^2	64.5	66.0	56.7	64.7	62.0	–	–
dilbert	Acc	96.9	98.0	94.5	93.2	97.9	–	–
gina-agnostic	Acc	93.8	94.4	81.3	81.4	92.8	–	–
gisette	Acc	97.5	97.7	95.2	98.1	97.0	–	–
internet-advertisements	Acc	96.8	97.0	96.3	**97.1**	96.3	–	–
madelon	Acc	70.8	64.4	52.7	56.2	59.2	–	–
mfeat-factors	Acc	97.8	97.2	98.0	97.5	98.0	–	–
ova-lung	Acc	98.7	98.1	97.7	**99.0**	98.4	–	–
qsar-tid-10980	R^2	66.0	40.9	53.5	42.7	64.0	–	–
qsar-tid-194	R^2	76.9	59.8	69.0	63.2	73.9	–	–
qsar-tid-72	R^2	59.3	39.0	49.5	44.5	58.4	–	–
santander-transaction-v.	R^2	18.2	11.1	−2.4	−19679277	3.9	–	–
semeion	Acc	94.7	95.3	89.3	94.4	97.8	–	–
tecator	R^2	98.7	99.5	85.8	99.6	94.7	–	–
topo-2-1	R^2	5.1	2.1	−12.4	0.8	0.7	–	–

Table 56: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 4 of 4 (Deep), fresh (26) and held-out (10) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 143 ok, 3 failed, 0 n/a, 430 missing.

Table		MLP	MLP-PLR	ResNet	SNN	DCN2	NODE	TabNet	AutoInt	FTT	SAINT	ExcelF	TabR	RTabR	MNCA	TabM	RealMLP
Fresh (26)																	
amazon-employee-access	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
anneal	Acc	97.7	98.7	99.0	92.7	97.0	83.7	76.0	96.3	97.7	97.7	99.3	99.0	98.7	99.0	97.7	**99.7**
apsfailure	Acc	–	–	99.2	–	–	–	–	–	–	–	–	–	–	–	–	–
bank-customer-churn	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bioresponse	Acc	77.1	75.4	75.3	–	–	–	–	–	×	–	–	77.8	78.3	–	79.5	79.5
churn	Acc	92.5	94.8	93.2	–	–	–	–	–	95.1	–	–	93.3	94.7	–	94.2	94.7
coil2000-insurance-poli.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
credit-g	Acc	71.9	71.0	75.1	–	–	–	–	–	75.1	–	–	76.0	73.7	–	74.0	77.5
customer-satisfaction-i.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
diabetes130us	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
e-commereshippingdata	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fitness-club	Acc	76.8	76.4	76.4	–	–	–	–	–	76.4	–	–	76.0	76.2	–	76.8	76.8
givemesomecredit	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hiva-agnostic	Acc	**96.5**	**96.5**	**96.5**	–	–	–	–	–	×	–	–	**96.5**	**96.5**	–	**96.5**	95.9
hr-analytics-job-change.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
in-vehicle-coupon-recom.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
is-this-a-good-customer	Acc	88.5	88.5	87.3	–	–	–	–	–	87.1	–	–	86.8	84.5	–	88.2	87.7
jm1	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
kddcup09-appetency	Acc	–	–	**98.2**	–	–	–	–	–	–	–	–	–	–	–	–	–
marketing-campaign	Acc	89.0	88.4	87.7	–	–	–	–	–	88.4	–	–	89.6	88.5	–	87.4	87.3
mic	Acc	88.4	88.7	88.0	–	–	–	–	–	88.4	–	–	87.5	87.3	–	88.5	87.8
naticusdroid	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
polish-companies-bankru.	Acc	94.2	94.1	94.2	–	–	–	–	–	94.8	–	–	93.8	93.4	–	94.1	95.4
sdss17	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
seismic-bumps	Acc	**93.4**	**93.4**	**93.4**	–	–	–	–	–	**93.4**	–	–	**93.4**	**93.4**	–	**93.4**	**93.4**
taiwanese-bankruptcy-pr.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Held-out (10)																	
another-dataset-on-used.	R^2	83.7	85.2	83.4	–	–	–	–	–	85.5	–	–	84.4	85.6	–	84.7	85.8
bank-marketing	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
diabetes	Acc	77.0	75.0	75.4	73.4	76.6	65.6	69.5	76.2	77.0	76.6	78.5	77.3	77.3	77.7	73.0	71.1
diamonds	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
food-delivery-time	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
hazelnut-spread-contami.	Acc	91.8	91.1	93.5	–	–	–	–	–	90.6	–	–	93.2	91.6	–	91.1	93.4
houses	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
online-shoppers-intenti.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
qsar-tid-11	R^2	68.3	64.6	68.0	–	–	–	–	–	×	–	–	70.1	70.8	–	69.8	69.5
splice	Acc	95.2	96.9	95.0	–	–	–	–	–	96.1	–	–	94.5	96.1	–	97.1	96.2

Table 57: Per-table test scores, full data, primary metric $\times 100$ (accuracy or R^2): methods block 4 of 4 (Deep), battery (15) and high-dim. (20) tables. Bold: best of all 48 methods on the table (all blocks). $\times$: failed (scored as the table's worst method in the aggregates); n/a: not applicable; –: no result; ‡: fitted on a row subsample of at most 10,000 rows. Cells in this block: 80 ok, 0 failed, 0 n/a, 480 missing.

Table		MLP	MLP-PLR	ResNet	SNN	DCN2	NODE	TabNet	AutoInt	FTT	SAINT	ExcelF	TabR	RTabR	MNCA	TabM	RealMLP
Battery (15)																	
airfoil-self-noise	R^2	96.0	96.7	94.6	–	–	–	–	–	95.4	–	–	96.2	95.8	–	96.3	96.3
blood-transfusion-servi.	Acc	77.2	76.0	78.4	80.4	80.4	76.8	74.4	**80.8**	78.4	**80.8**	78.8	78.8	80.4	79.2	79.6	78.4
concrete-compressive-st.	R^2	90.7	91.9	88.9	–	–	–	–	–	89.3	–	–	90.3	89.0	–	91.0	90.5
credit-card-clients-def.	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
healthcare-insurance-ex.	R^2	85.5	86.6	84.8	–	–	–	–	–	84.6	–	–	85.0	85.7	–	86.4	86.4
heloc	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
maternal-health-risk	Acc	71.3	72.8	67.8	–	–	–	–	–	71.0	–	–	79.6	79.9	–	74.6	74.3
miami-housing	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
physiochemical-protein	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
qsar-biodeg	Acc	84.4	84.7	86.1	–	–	–	–	–	84.4	–	–	86.6	87.2	–	83.2	86.9
qsar-fish-toxicity	R^2	60.0	54.2	58.5	–	–	–	–	–	51.4	–	–	60.1	61.3	–	59.5	59.9
students-dropout-and-ac.	Acc	75.6	76.2	74.2	–	–	–	–	–	76.0	–	–	75.7	76.9	–	76.1	76.8
superconductivity	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
website-phishing	Acc	88.0	89.6	91.6	–	–	–	–	–	90.5	–	–	91.4	91.1	–	89.6	90.2
wine-quality	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
High-dim. (20)																	
ap-breast-ovary	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ap-colon-kidney	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
arcene	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
christine	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
cnae-9	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
communities-and-crime	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dilbert	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
gina-agnostic	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
gisette	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
internet-advertisements	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
madelon	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mfeat-factors	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ova-lung	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
qsar-tid-10980	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
qsar-tid-194	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
qsar-tid-72	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
santander-transaction-v.	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
semeion	Acc	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
tecator	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
topo-2-1	R^2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

I.4 REGISTERED FAILURES-IMPUTED ANALYSIS AND THE FRESH REGRESSION GROUP

The whole field. Table 58 extends Table 1 to every configuration that has cells in at least one of its three groups, including the unbagged twins of the bagged models and the methods that were not run on the regression group.

Table 58: **Every configuration on the three groups of Table 1.** Columns and marks as in Table 1; –: not run on that group.

	Fresh TabArena classification (22 tables)		Held-out (10 tables)		Fresh regression (23 tables)	
Method	Score	$\Delta_{[95\%\ CI]}$	Score	$\Delta_{[95\%\ CI]}$	Score	$\Delta_{[95\%\ CI]}$
TabEvo	88.3$_{[84.6, 91.8]}$		82.5$_{[69.8, 91.6]}$		71.3$_{[61.0, 81.0]}$	
Foundation models						
TabFM	**89.0**	−0.66$_{[−1.03, −0.31]}$	84.1	−1.64$_{[−3.04, −0.39]}$	**72.6**	−1.35$_{[−2.45, −0.42]}$
TabPFN v3	88.7	−0.42$_{[−0.73, −0.13]}$	83.5	−0.97$_{[−1.84, −0.03]}$	72.4	−1.12$_{[−1.87, −0.49]}$
TabPFN v2.5	88.7	−0.36$_{[−0.66, −0.06]}$	82.9	−0.39$_{[−1.08, 0.32]}$	72.1	−0.82$_{[−1.43, −0.29]}$
TabICLv2	88.6	−0.29$_{[−0.62, 0.04]}$	82.6	−0.08$_{[−1.01, 0.97]}$	70.6	+0.71$_{[−0.82, 2.95]}$
TabDPT	88.6	−0.23$_{[−0.60, 0.17]}$	83.2	−0.67$_{[−1.15, −0.22]}$	–	
LimiX	88.5[20]	−0.85$_{[−1.22, −0.48]}$	**84.8**	−2.28$_{[−3.54, −1.21]}$	–	
TabICL v1.1	88.5	−0.15$_{[−0.45, 0.14]}$	89.5[5]	−0.02$_{[−1.13, 1.10]}$	–	
TabPFN v2	88.4	−0.03$_{[−0.42, 0.43]}$	82.0	+0.46$_{[−0.61, 1.69]}$	71.3	−0.08$_{[−0.72, 0.68]}$
Mitra	88.0[16]	−0.20$_{[−0.66, 0.26]}$	88.7[5]	+0.23$_{[−0.65, 1.32]}$	–	
Gradient boosting and stacking						
Stack-TD	88.5	−0.17$_{[−0.56, 0.15]}$	82.3	+0.19$_{[−0.78, 1.02]}$	70.6	+0.64$_{[0.09, 1.26]}$
CatBoost-TD$_{b5}$	88.3	+0.01$_{[−0.26, 0.24]}$	82.2	+0.25$_{[−0.86, 1.21]}$	69.6	+1.67$_{[0.76, 2.68]}$
CatBoost-TD	88.2	+0.18$_{[−0.09, 0.44]}$	81.7	+0.80$_{[−0.49, 2.09]}$	68.3	+2.92$_{[1.86, 4.07]}$
CatBoost-D	88.1	+0.22$_{[−0.07, 0.53]}$	81.2	+1.34$_{[−0.30, 3.06]}$	69.6	+1.70$_{[0.82, 2.74]}$
Ensemble-TD	88.1	+0.25$_{[−0.12, 0.61]}$	81.7	+0.84$_{[−0.52, 2.40]}$	70.5	+0.76$_{[0.12, 1.46]}$
LightGBM-TD$_{b5}$	88.0	+0.32$_{[0.02, 0.62]}$	81.9	+0.61$_{[−0.14, 1.43]}$	69.6	+1.66$_{[1.05, 2.38]}$
HistGB-D	87.9	+0.43$_{[0.08, 0.80]}$	81.4	+1.09$_{[−0.48, 2.57]}$	–	
XGBoost-TD	87.9	+0.43$_{[0.00, 0.89]}$	80.2	+2.28$_{[0.66, 4.40]}$	67.8	+3.49$_{[2.51, 4.51]}$
LightGBM-TD	87.9	+0.44$_{[0.07, 0.80]}$	81.2	+1.29$_{[0.33, 2.34]}$	68.3	+2.95$_{[2.07, 3.84]}$
LightGBM-D	87.8	+0.50$_{[0.15, 0.91]}$	80.9	+1.55$_{[0.39, 2.80]}$	69.0	+2.29$_{[1.43, 3.25]}$
XGBoost-TD$_{b5}$	87.8	+0.53$_{[0.16, 0.93]}$	81.0	+1.49$_{[0.16, 3.41]}$	69.1	+2.20$_{[1.45, 3.00]}$
XGBoost-D	87.8	+0.54$_{[0.19, 0.92]}$	79.7	+2.82$_{[0.91, 5.08]}$	66.6	+4.64$_{[3.16, 6.17]}$
Random forest	87.6	+0.75$_{[0.38, 1.13]}$	80.6	+1.89$_{[0.90, 3.05]}$	67.2	+4.02$_{[2.59, 5.55]}$
ExtraTrees-D	87.3	+1.01$_{[0.51, 1.55]}$	77.1	+5.35$_{[1.17, 10.62]}$	–	
Neural networks						
RealMLP-TD$_{b5}$	88.3	+0.02$_{[−0.23, 0.26]}$	81.4	+1.12$_{[0.14, 2.29]}$	70.4	+0.85$_{[−0.13, 2.09]}$
FT-Transformer-D	88.2[21]	+0.51$_{[0.07, 1.03]}$	79.9	+2.62$_{[0.70, 5.36]}$	67.6[22]	+3.26$_{[1.58, 5.24]}$
TabM-D$_{b5}$	88.0	+0.31$_{[−0.05, 0.68]}$	81.1	+1.37$_{[0.24, 2.85]}$	69.9	+1.39$_{[0.35, 2.66]}$
TabM-D	87.8	+0.49$_{[0.08, 0.90]}$	80.4	+2.08$_{[0.73, 3.51]}$	69.5	+1.78$_{[0.70, 3.26]}$
RealMLP-TD	87.8	+0.51$_{[0.06, 0.98]}$	80.3	+2.16$_{[0.70, 3.80]}$	69.1	+2.18$_{[0.89, 3.73]}$
TabR-D	87.6	+0.74$_{[0.22, 1.27]}$	81.4	+1.11$_{[0.49, 1.76]}$	68.4	+2.90$_{[1.81, 4.09]}$
RealTabR-D	87.6	+0.76$_{[0.25, 1.31]}$	81.2	+1.32$_{[0.38, 2.63]}$	70.0	+1.30$_{[0.68, 1.96]}$
AutoInt	87.5[21]	+1.21$_{[0.40, 2.15]}$	80.1[9]	+3.48$_{[1.36, 5.80]}$	–	
MLP-PLR-D	87.4	+0.94$_{[0.24, 1.75]}$	79.9	+2.55$_{[1.02, 4.27]}$	69.0	+2.26$_{[1.05, 3.85]}$
MLP-D	87.4	+0.94$_{[0.29, 1.65]}$	80.7	+1.84$_{[0.87, 2.82]}$	67.1	+4.15$_{[2.34, 6.55]}$
ResNet-D	87.4	+0.96$_{[0.37, 1.63]}$	80.6	+1.92$_{[0.89, 3.09]}$	67.2	+4.03$_{[2.47, 5.97]}$
DCNv2	87.0	+1.31$_{[0.59, 2.09]}$	71.7	+10.81$_{[1.19, 28.47]}$	–	
ModernNCA	86.7[21]	+1.08$_{[0.37, 1.91]}$	79.9	+2.56$_{[0.68, 5.29]}$	–	
ExcelFormer	86.5[18]	+1.14$_{[0.37, 2.10]}$	81.7[9]	+1.90$_{[0.45, 3.49]}$	–	
SNN	85.9	+2.39$_{[1.14, 3.75]}$	62.0	+20.50$_{[8.06, 34.85]}$	–	
TabNet	84.7	+3.62$_{[1.71, 6.11]}$	74.6	+7.92$_{[3.27, 13.69]}$	–	
NODE	84.6[21]	+4.05$_{[1.97, 6.43]}$	68.0	+14.53$_{[6.77, 24.68]}$	–	
SAINT	83.3[16]	+3.50$_{[1.54, 5.99]}$	81.3[9]	+2.28$_{[1.00, 3.71]}$	–	
Classical						
GP (ARD, exact)	87.5	+0.79$_{[0.31, 1.31]}$	80.2	+2.32$_{[0.70, 4.49]}$	67.7	+3.54$_{[1.10, 6.74]}$
SVM	87.2	+1.11$_{[0.51, 1.76]}$	79.4	+3.13$_{[0.98, 5.56]}$	–	
xRFM-D	86.9[21]	+1.28$_{[0.73, 1.86]}$	80.8	+1.73$_{[0.52, 3.07]}$	–	

Failures imputed (registered primary analysis of the fresh group). Table 59 scores all 26 fresh classification tables, the three on which TabEvo has no result (amazon-employee-access, kddcup09-appetency, sdss17) counted as the worst method, as the protocol prescribes. Under this rule no parity is established with any foundation model: the program is 0.9 to 1.5 points behind seven of the eight foundation models of Table 1 (significantly behind TabFM, LimiX and TabPFN v2.5) and ahead of Mitra, which fails on ten of the 26; it is 0.04 to 0.92 points behind the trained models (none significantly), 1.1 behind Stack-TD, which completes every table, and 0.7 ahead of the exact ARD GP.

Table 59: **Fresh classification, all 26 tables, failures imputed as the worst method.** $\Delta = TabEvo -$ method in points with its 95 % bootstrap interval; p: Wilcoxon signed-rank, Holm-corrected over the rows; marks as in Table 1.

	Fresh TabArena, all 26 tables	
TabEvo vs.	Δ [95% CI]	p_{Holm}
Foundation models		
TabFM	$-1.47_{[-2.95, -0.54]}$	.007
TabPFN v3	$-1.27_{[-2.77, -0.35]}$	.074
TabPFN v2.5	$-1.21_{[-2.70, -0.29]}$	.031
TabICLv2	$-1.16_{[-2.67, -0.22]}$	.187
TabDPT	$-1.10_{[-2.63, -0.12]}$	.605
LimiX	$-1.51_{[-3.05, -0.54]}$	.016
TabPFN v2	$-0.90_{[-2.45, 0.11]}$	.524
Mitra	$+0.53_{[-0.12, 1.29]}$	1.000
Gradient boosting and stacking		
Stack-TD	$-1.06_{[-2.56, -0.09]}$	1.000
CatBoost-TD$_{\text{b5}}$	$-0.92_{[-2.44, 0.03]}$	1.000
CatBoost-D	$-0.71_{[-2.23, 0.26]}$	1.000
Ensemble-TD	$-0.69_{[-2.27, 0.32]}$	1.000
LightGBM-TD$_{\text{b5}}$	$-0.64_{[-2.19, 0.35]}$	1.000
LightGBM-D	$-0.47_{[-2.06, 0.57]}$	1.000
XGBoost-TD$_{\text{b5}}$	$-0.40_{[-1.90, 0.60]}$	1.000
XGBoost-D	$-0.42_{[-1.99, 0.60]}$	1.000
Random forest	$-0.26_{[-1.87, 0.80]}$	1.000
Neural networks		
RealMLP-TD$_{\text{b5}}$	$-0.84_{[-2.30, 0.05]}$	1.000
FT-Transformer-D	$-0.04_{[-1.66, 1.18]}$	1.000
TabM-D$_{\text{b5}}$	$-0.56_{[-2.00, 0.38]}$	1.000
TabR-D	$-0.22_{[-1.77, 0.87]}$	1.000
RealTabR-D	$-0.18_{[-1.68, 0.89]}$	1.000
MLP-PLR-D	$-0.05_{[-1.70, 1.21]}$	1.000
MLP-D	$-0.05_{[-1.68, 1.15]}$	1.000
ResNet-D	$-0.04_{[-1.65, 1.14]}$	1.000
Classical		
GP (ARD, exact)	$+0.67_{[0.25, 1.12]}$	.119

Fresh regression group. Twenty-four regression tables from the TabM, TabReD and Grinsztajn suites that belong to no other group (the selection rule and the list are in Table 5, Amendment 17), evaluated in the 4,000-row regime with the test split capped at 20,000 rows for every method, one seed, one hour per cell. Table 60 gives the aggregate on the tables TabEvo completes and on all 24 with failures imputed, with the pre-registered primary comparison against Stack-TD and the two attribution controls (the exact ARD GP alone; Stack-TD with that GP as a fourth base learner); Table 61 gives every cell. The pre-registered criterion for a practical advantage over Stack-TD (all 24 tables with failures imputed; interval above zero, effect of at least one point, $p < .05$) is not met: $\Delta = +0.40$ [-0.31, $+1.10$], $p_{\text{Holm}} = 0.282$; on the 23 tables the program completes it is ahead by $+0.64$ [$+0.09$, $+1.26$] ($p_{\text{Holm}} = 0.098$).

Table 60: **Fresh regression (24 tables), aggregate.** $R^2 \times 100$; $\Delta = TabEvo -$ method with its 95 % bootstrap interval; p: Wilcoxon signed-rank, Holm-corrected over the rows; W/T/L from TabEvo's side; marks as in Table 1. Left: tables TabEvo completes; right: all 24, failures imputed as the worst method.

	23 tables TabEvo completes				all 24, failures as worst	
Method	Score	Δ [95% CI]	p_{Holm}	W/T/L	Δ [95% CI]	p_{Holm}
TabEvo	$71.3_{[61.0, 81.0]}$				$68.6_{[57.4, 79.1]}$	
TabFM	72.6	$-1.35_{[-2.45, -0.42]}$	.015	5/0/18	$-1.61_{[-2.78, -0.58]}$	.012
TabPFN v3	72.4	$-1.12_{[-1.87, -0.49]}$	.003	4/0/19	$-1.30_{[-2.13, -0.61]}$	.002
TabPFN v2.5	72.1	$-0.82_{[-1.43, -0.29]}$	.003	2/0/21	$-0.84_{[-1.44, -0.34]}$	.002
TabPFN v2	71.3	$-0.08_{[-0.72, 0.68]}$	.629	8/0/15	$-0.19_{[-0.87, 0.55]}$	.436
TabICLv2	70.6	$+0.71_{[-0.82, 2.95]}$	.629	11/0/12	$+0.40_{[-1.19, 2.66]}$	.436
Stack-TD + GP	72.7[1]	$+0.24_{[-0.28, 0.63]}$	.098	14/0/3	$+2.14_{[0.21, 4.79]}$	.035
Stack-TD	70.6	$+0.64_{[0.09, 1.26]}$	.098	17/0/6	$+0.40_{[-0.31, 1.10]}$	.282
Ensemble-TD	70.5	$+0.76_{[0.12, 1.46]}$	.098	17/0/6	$+0.54_{[-0.23, 1.29]}$	.282
LightGBM-TD$_{\text{b5}}$	69.6	$+1.66_{[1.05, 2.38]}$	<.001	23/0/0	$+1.37_{[0.52, 2.21]}$	.001
CatBoost-TD$_{\text{b5}}$	69.6	$+1.67_{[0.76, 2.68]}$	.003	19/0/4	$+1.36_{[0.29, 2.47]}$	.028
CatBoost-D	69.6	$+1.70_{[0.82, 2.74]}$	<.001	21/0/2	$+1.44_{[0.44, 2.55]}$	.006
XGBoost-TD$_{\text{b5}}$	69.1	$+2.20_{[1.45, 3.00]}$	<.001	22/0/1	$+1.88_{[0.92, 2.83]}$	.001
LightGBM-D	69.0	$+2.29_{[1.43, 3.25]}$	<.001	22/0/1	$+2.15_{[1.30, 3.13]}$	<.001
CatBoost-TD	68.3	$+2.92_{[1.86, 4.07]}$	<.001	23/0/0	$+2.66_{[1.52, 3.90]}$	<.001
LightGBM-TD	68.3	$+2.95_{[2.07, 3.84]}$	<.001	23/0/0	$+2.73_{[1.82, 3.71]}$	<.001
XGBoost-TD	67.8	$+3.49_{[2.51, 4.51]}$	<.001	23/0/0	$+3.21_{[2.12, 4.32]}$	<.001
Random forest	67.2	$+4.02_{[2.59, 5.55]}$	<.001	22/0/1	$+3.69_{[2.22, 5.30]}$	<.001
XGBoost-D	66.6	$+4.64_{[3.16, 6.17]}$	<.001	22/0/1	$+4.45_{[3.02, 6.02]}$	<.001
RealMLP-TD$_{\text{b5}}$	70.4	$+0.85_{[-0.13, 2.09]}$	.098	17/0/6	$+0.61_{[-0.44, 1.85]}$	.282
RealTabR-D	70.0	$+1.30_{[0.68, 1.96]}$	.006	18/0/5	$+1.09_{[0.34, 1.81]}$	.035
TabM-D$_{\text{b5}}$	69.9	$+1.39_{[0.35, 2.66]}$	.034	18/0/5	$+1.06_{[-0.13, 2.44]}$	.127
TabM-D	69.5	$+1.78_{[0.70, 3.26]}$	.003	19/0/4	$+1.46_{[0.23, 3.06]}$	.028
RealMLP-TD	69.1	$+2.18_{[0.89, 3.73]}$	.008	18/0/5	$+2.00_{[0.71, 3.52]}$	.032
MLP-PLR-D	69.0	$+2.26_{[1.05, 3.85]}$	<.001	20/0/3	$+1.98_{[0.68, 3.60]}$	.006
TabR-D	68.4	$+2.90_{[1.81, 4.09]}$	<.001	21/0/2	$+2.59_{[1.38, 3.83]}$	.006
FT-Transformer-D	67.6[2]	$+3.26_{[1.58, 5.24]}$	<.001	20/0/2	$+2.93_{[1.31, 4.82]}$	.002
ResNet-D	67.2	$+4.03_{[2.47, 5.97]}$	<.001	21/0/2	$+3.74_{[2.14, 5.73]}$	<.001
MLP-D	67.1	$+4.15_{[2.34, 6.55]}$	<.001	23/0/0	$+3.83_{[1.97, 6.34]}$	<.001
GP (ARD, exact)	67.7	$+3.54_{[1.10, 6.74]}$	<.001	23/0/0	$+3.27_{[0.92, 6.39]}$	<.001
TabEvo seed	70.7	$+0.60_{[0.12, 1.44]}$	<.001	20/2/1	$+0.58_{[0.12, 1.40]}$	<.001

Table 61: **Fresh regression, every cell.** Test $R^2 \times 100$ (raw; negative values shown); ×: failed or not finished within the limit. Bold: best of the row.

	5g-energy-consu.	*black-friday*	*cooking-time*	*delivery-eta*	*house*	*maps-routing*	*microsoft*	*nyc-taxi-green-.*	*particulate-mat.*	*Brazilian_houses*	*Mercedes_Benz_G.*	*OnlineNewsPopul.*
TabEvo	90.5	50.0	47.0	41.7	60.4	79.4	×	54.0	67.5	99.5	47.0	14.6
TabEvo seed	90.6	50.0	46.9	41.2	59.9	79.2	×	52.9	67.4	99.5	46.8	14.0
TabFM	90.4	**50.7**	46.9	41.5	67.2	80.0	**15.5**	**63.4**	68.9	99.9	**47.8**	**16.3**
TabPFN v3	90.6	50.3	46.4	41.8	**67.6**	79.3	13.5	55.4	**69.1**	99.4	47.6	16.0
TabPFN v2.5	**91.0**	49.7	**47.5**	41.9	65.2	**80.0**	9.4	52.7	68.0	**100.0**	47.4	15.1
TabPFN v2	90.9	49.6	47.3	41.4	62.8	79.8	11.0	50.0	68.4	100.0	47.4	15.2
TabICLv2	90.5	50.3	45.9	41.0	66.0	79.3	14.8	47.6	68.6	99.9	47.8	15.9
Stack-TD + GP	×	49.5	×	×	59.4	×	13.5	52.4	66.5	99.8	47.4	14.6
Stack-TD	90.1	48.3	46.0	**42.7**	59.3	79.2	13.1	49.5	66.4	99.8	47.3	14.4
Ensemble-TD	89.8	47.7	46.0	42.5	57.3	79.3	12.7	48.9	66.3	99.8	47.3	13.9
LightGBM-TD$_{b5}$	90.0	48.1	45.2	40.6	58.9	79.2	13.4	51.4	65.6	99.5	46.2	14.3
CatBoost-TD$_{b5}$	90.1	48.3	45.7	41.0	56.0	78.8	13.7	49.3	66.8	99.8	46.6	14.2
CatBoost-D	90.3	48.0	45.3	41.5	59.0	79.1	12.7	45.9	66.6	99.8	47.0	13.3
XGBoost-TD$_{b5}$	89.2	47.1	45.3	40.5	58.5	79.2	13.5	49.8	65.3	99.5	46.0	14.4
LightGBM-D	89.6	47.2	44.5	40.1	57.7	78.6	9.2	49.4	65.9	99.7	46.3	12.8
CatBoost-TD	89.9	47.2	43.4	40.0	54.0	78.4	11.3	46.4	66.0	99.5	45.8	11.0
LightGBM-TD	89.7	46.2	43.1	39.2	57.7	78.5	10.3	48.4	64.4	99.4	44.6	12.2
XGBoost-TD	88.9	45.6	43.1	39.0	55.5	78.4	11.4	45.5	63.9	99.3	44.9	11.3
Random forest	88.9	42.4	44.5	39.7	54.9	78.8	11.9	49.7	62.3	99.7	42.7	13.1
XGBoost-D	89.2	47.4	41.9	37.1	54.7	76.8	8.1	44.3	63.8	99.7	45.0	9.4
RealMLP-TD$_{b5}$	90.1	46.5	46.8	41.7	59.5	79.0	12.9	42.1	65.3	99.8	47.6	13.1
RealTabR-D	89.7	46.8	45.5	41.0	57.1	78.1	11.9	51.6	64.5	99.7	47.2	10.5
TabM-D$_{b5}$	89.9	40.4	47.0	41.9	61.5	79.0	14.6	44.3	65.5	99.8	46.6	13.7
TabM-D	89.7	35.9	46.5	42.1	61.1	78.8	14.0	45.2	65.2	99.8	45.9	11.7
RealMLP-TD	89.8	46.9	45.4	42.6	53.3	77.7	10.2	39.2	60.0	99.6	47.2	12.8
MLP-PLR-D	90.1	39.2	46.6	42.5	60.0	78.4	12.5	39.2	64.6	99.8	45.8	11.6
TabR-D	90.2	45.1	41.2	30.7	56.4	77.1	12.6	47.5	65.1	99.7	45.1	11.5
FT-Transformer-D	89.8	38.7	45.4	41.7	58.3	×	12.0	40.2	64.2	99.8	46.4	12.4
ResNet-D	90.0	30.1	43.3	37.4	60.9	77.3	11.1	43.9	63.9	99.6	44.4	10.4
MLP-D	89.6	25.5	44.4	36.7	58.4	77.7	11.7	38.4	64.3	99.5	45.3	11.3
GP (ARD, exact)	90.1	50.0	46.5	41.4	35.6	78.0	10.8	47.0	64.7	99.2	46.7	10.8

	analcatdata_sup.	*house_sales*	*year*	*Ailerons*	*cpu_act*	*elevators*	*fifa*	*isolet*	*medical_charges*	*pol*	*sberbank-housing*	*weather*
TabEvo	98.4	89.8	29.3	85.2	98.9	92.4	64.7	90.3	97.8	99.4	59.0	82.3
TabEvo seed	98.1	89.7	29.2	85.1	98.8	92.4	64.3	90.3	97.8	99.4	49.6	82.1
TabFM	98.5	**90.3**	**31.9**	86.3	**99.0**	93.0	**66.5**	**95.9**	97.9	99.3	56.3	82.4
TabPFN v3	98.3	90.2	31.2	**86.4**	98.9	**93.2**	66.0	94.5	97.9	**99.6**	62.7	**82.5**
TabPFN v2.5	**98.5**	90.2	31.0	85.8	98.9	92.9	65.6	93.8	97.9	99.4	**63.3**	82.4
TabPFN v2	98.3	90.2	30.5	85.3	98.1	91.9	65.8	93.2	97.9	94.5	61.6	80.8
TabICLv2	98.3	89.7	31.5	85.6	98.8	92.9	65.9	92.2	**97.9**	98.8	37.4	81.3
Stack-TD + GP	98.4	89.1	28.2	84.6	98.8	92.1	63.9	93.4	97.8	99.3	×	×
Stack-TD	98.2	88.4	25.4	84.3	98.5	91.8	63.8	93.0	97.8	99.2	59.4	81.5
Ensemble-TD	98.1	88.3	25.5	84.4	98.6	91.8	63.9	93.0	97.8	99.3	60.6	81.5
LightGBM-TD$_{b5}$	98.2	88.1	27.2	83.7	98.0	86.3	63.7	83.8	97.6	98.5	56.3	80.8
CatBoost-TD$_{b5}$	98.4	88.3	25.5	83.9	96.8	86.7	64.7	81.2	97.7	98.5	61.1	81.3
CatBoost-D	98.2	88.6	25.1	84.2	98.3	87.9	64.7	81.6	97.8	97.7	59.1	81.0
XGBoost-TD$_{b5}$	98.1	87.7	25.0	81.8	97.2	85.6	63.5	83.3	97.5	98.4	55.2	80.5
LightGBM-D	98.1	87.7	23.5	83.2	98.5	85.2	63.5	81.6	97.7	98.1	58.0	79.7
CatBoost-TD	98.1	87.5	21.8	83.2	96.9	85.5	63.5	80.4	97.7	98.3	57.2	80.5
LightGBM-TD	98.0	87.1	23.8	82.7	98.4	84.8	62.2	82.8	97.4	98.3	52.8	79.6
XGBoost-TD	97.8	86.6	21.8	81.6	97.4	83.7	62.0	82.9	97.3	98.0	54.8	79.6
Random forest	98.3	85.9	20.1	82.3	98.0	78.9	61.9	79.7	97.5	98.0	50.2	78.9
XGBoost-D	97.8	86.7	19.6	80.1	96.7	83.6	62.4	77.6	97.5	97.3	45.6	78.2
RealMLP-TD$_{b5}$	98.3	88.6	28.8	84.1	98.4	92.1	63.5	94.7	97.9	99.4	61.0	81.6
RealTabR-D	98.3	88.3	24.9	83.5	98.5	92.0	62.7	92.6	97.8	99.1	59.3	80.5
TabM-D$_{b5}$	98.2	88.8	28.9	84.2	98.7	91.4	63.9	92.8	97.7	98.5	53.4	81.4
TabM-D	98.0	88.3	27.1	84.1	98.6	90.9	64.1	91.1	97.7	98.4	56.9	81.0
RealMLP-TD	98.1	87.4	23.4	84.0	98.2	91.8	62.4	92.9	97.8	99.2	58.0	81.2
MLP-PLR-D	98.3	88.3	26.7	83.8	98.2	91.3	62.8	85.9	97.8	99.1	56.7	80.4
TabR-D	98.1	87.4	22.9	82.5	98.5	90.4	63.2	92.9	97.7	95.8	55.8	77.8
FT-Transformer-D	98.2	87.6	21.7	83.8	98.5	91.5	63.8	75.8	97.8	99.3	53.5	79.8
ResNet-D	97.5	86.9	23.9	82.4	98.3	90.1	63.2	86.4	97.2	93.0	48.4	77.7
MLP-D	97.8	86.0	23.8	82.6	97.5	89.8	63.7	88.7	97.2	93.7	53.0	78.6
GP (ARD, exact)	98.4	88.3	26.7	83.7	98.7	91.8	62.6	88.2	97.7	98.1	32.0	81.6

I.5 COVERAGE AND FAILED CELLS

Failure policy. Each (method, table) cell is one fit with a two-hour limit. A fit that raises an error or exceeds the limit is a recorded failure, never a silent skip: in every aggregate it is scored as the table's worst method (TabArena imputes a default random forest instead), and the complete-case variant drops every table with a failed cell. A classification-only model (TabICL v1.1) on a regression table is *not applicable*: the cell is excluded from that model's statistics, not counted as a failure. Cells of the main-text methods that have no result are scored as failures in the imputed statistics of the main text. For TabEvo they are the fresh tables amazon-employee-access, kddcup09-appetency and sdss17, on which its fits did not finish within the two-hour limit: reruns under the same rule were killed at 7,200 s after one-hot encoding had produced 9,655, 27,616 and 5,095 columns (its fit on diabetes130us finished in 4,721 s and is included); on the high-dimensional tables they are five timeouts (ap-breast-ovary, ap-colon-kidney, ova-lung, dilbert, gisette). The pruned program speed A completes sdss17 and amazon-employee-access within the limit and times out on kddcup09-appetency. Our recorded failures are exact GP fits whose Gram matrix is not positive definite on wide sparse-binary tables (hiva-agnostic, 1,617 categorical features; internet-advertisements, 1,555 of its 1,558 features categorical), and one out-of-memory fit of the seed on amazon-employee-access.

Table 62: Runtime of every method, full data, ok cells: median fit, predict and fit+predict seconds per TabArena table, the 90th percentile of fit+predict, total fit+predict hours over the TabArena tables, and the median fit+predict seconds on the high-dimensional tables. n = TabArena tables timed. HW: GPU = one NVIDIA H100 80 GB slot of a shared pool (8 GPUs), CPU = 8 threads of a shared 96-core host; – = not recorded. Times were measured on the shared pool while other benchmark jobs ran and include that contention.

Method	Family	HW	n	Fit	Predict	Fit+pred.	p90	Total (h)	n hd	Fit+pred. hd
TabEvo (full data)	Ours	GPU	2	660	14	674	859	0.4	–	–
TabEvo evolved ($\leq$10k rows)	Ours	GPU	46	644	1.8	646	1,797	11.7	12	1,240
TabEvo seed ($\leq$10k rows)	Ours	GPU	45	508	1.2	510	1,644	10.1	13	837
TabEvo speed-up A	Ours	GPU	23	48	1.4	49	241	0.7	–	–
TabEvo speed-up B	Ours	GPU	23	26	1.1	34	228	0.6	–	–
TabICL v1.1	FM	GPU	38	11	2.1	15	46	0.3	–	–
TabICLv2	FM	GPU	51	13	1.6	20	46	0.4	16	37
TabPFN v2	FM	GPU	51	13	2.1	15	31	0.3	–	–
TabPFN v2.5	FM	GPU	51	6.0	2.8	14	32	0.3	–	–
TabPFN v3	FM	GPU	51	12	2.9	17	40	0.3	16	24
TabFM	FM	GPU	51	28	25	66	342	1.9	16	132
Mitra	FM	GPU	4	30	6.3	37	100	0.1	–	–
TabDPT	FM	GPU	4	6.5	14	20	21	0.0	–	–
LimiX	FM	GPU	4	5.0	37	41	50	0.0	–	–
XGBoost (D)	GBDT	CPU	51	1.2	0.03	1.4	3.2	0.0	20	3.5
XGBoost (TD)	GBDT	CPU	51	2.2	0.04	2.3	9.3	0.1	20	9.7
LightGBM (D)	GBDT	CPU	51	1.3	0.03	1.5	2.7	0.0	20	2.2
LightGBM (TD)	GBDT	CPU	51	3.0	0.04	3.1	6.5	0.0	20	5.0
CatBoost (D)	GBDT	CPU	51	4.4	0.03	4.4	33	0.3	20	19
CatBoost (TD)	GBDT	CPU	51	2.8	0.03	2.9	15	0.1	20	29
CatBoost (TD, bag 5)	GBDT	CPU	51	21	0.14	21	110	0.9	20	605
LightGBM (TD, bag 5)	GBDT	CPU	51	18	0.21	18	58	0.7	20	24
XGBoost (TD, bag 5)	GBDT	CPU	51	16	0.28	16	111	0.7	20	54
HistGB (D)	GBDT	–	51	1.2	0.03	1.3	2.2	0.0	20	4.0
Ensemble (TD)	GBDT	GPU	25	12	0.17	12	80	0.2	–	–
RF (D)	Classical	CPU	51	1.1	0.06	1.2	3.5	0.0	20	1.5
ExtraTrees (D)	Classical	–	51	0.83	0.05	0.89	2.0	0.0	20	0.87
k-NN	Classical	–	51	0.00	0.19	0.19	3.7	0.1	20	0.25
Linear	Classical	–	51	0.02	0.11	0.14	0.49	0.0	20	0.22
SVM	Classical	–	51	2.8	0.33	3.2	35	0.2	20	7.2
GP (ARD)	Classical	GPU	5	130	23	138	1,985	1.0	–	–
xRFM (D)	Classical	GPU	4	7.7	16	21	38	0.0	–	–
MLP (D)	Deep	GPU	25	4.1	0.05	4.1	19	0.1	–	–
MLP-PLR (D)	Deep	GPU	25	4.3	0.06	4.4	22	0.1	–	–
ResNet (D)	Deep	GPU	27	4.8	0.06	5.1	24	0.1	–	–
SNN	Deep	GPU	3	6.1	0.22	6.3	16	0.0	–	–
DCNv2	Deep	GPU	3	4.0	0.22	4.2	20	0.0	–	–
NODE	Deep	GPU	3	5.4	0.23	5.6	11	0.0	–	–
TabNet	Deep	GPU	3	7.8	0.42	8.2	11	0.0	–	–
AutoInt	Deep	GPU	3	6.7	0.27	7.0	8.5	0.0	–	–
FT-Transformer (D)	Deep	GPU	22	2.7	0.04	2.7	11	0.0	–	–
SAINT	Deep	GPU	3	13	0.89	14	358	0.1	–	–
ExcelFormer	Deep	GPU	3	7.9	0.32	8.2	20	0.0	–	–
TabR (D)	Deep	GPU	25	13	0.09	14	60	0.2	–	–
RealTabR (D)	Deep	GPU	25	14	0.09	15	40	0.1	–	–
ModernNCA	Deep	GPU	3	4.5	0.22	4.7	19	0.0	–	–
TabM (D)	Deep	GPU	25	1.1	0.03	1.1	4.5	0.0	–	–
RealMLP (TD)	Deep	GPU	25	7.5	0.21	7.6	22	0.1	–	–

I.6 RUNTIME

Table 62 gives the runtime of every method and Figure 26 sets each method's mean normalised score against its median time. TabEvo fits four exact GPs, a learned linear metric and three CatBoost or MLP lanes per table, each inside a cross-fitting protocol, so its per-table cost is that of a small ensemble with hyperparameter searches rather than of one in-context forward pass; the pruned programs of Appendix F reduce it.

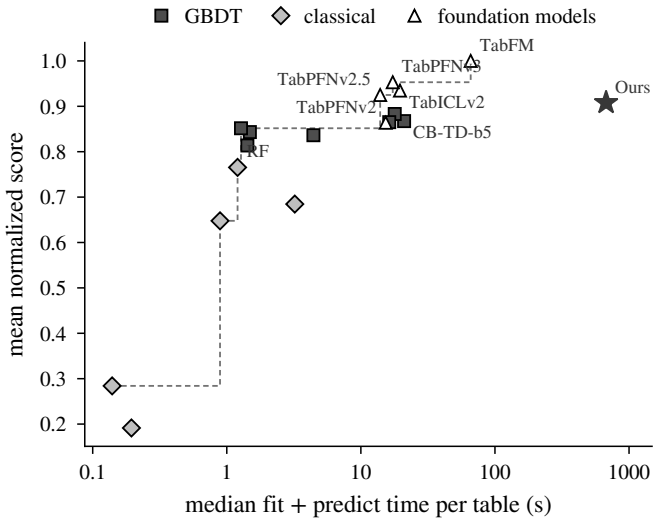


Figure 26: Mean normalised score against median fit+predict time per table, all TabArena tables, full data; the dashed step is the Pareto front. Marker shape encodes the family; TabEvo is the star (labelled Ours). Hardware per method in Table 62.

I.7 One-off cost: pretraining versus search

A foundation model is pretrained once and then applied to every table; our architecture is searched once, on the 15 battery tables, and then fixed. Table 63 sets the pretraining compute that the foundation models' authors report against the cost of our search. Where a source states it, the final pretraining run took 2,688 GPU-hours per TabPFN v2 checkpoint (8 RTX 2080 Ti for about two weeks), at least 1,440 for TabICL v1 (3 A100 for 20 days), 588 per TabICLv2 checkpoint (H100) and 480 for Mitra (8 A100 for about 60 hours); TabPFN v2.5, TabPFN v3, TabFM, TabDPT and LimiX do not report it. Our search took 11.7 h of wall-clock on a shared pool of 8 H100 GPUs (94 GPU-hours reserved), of which an estimated 46 GPU-hours were used (upper bound 86), plus 407 LLM calls with 17.8 M prompt and 0.82 M completion tokens (18.6 M in total; Appendix D). The two kinds of cost are not directly comparable: GPU types differ, and the development compute of every method, including the design of our hand-written seed, is excluded throughout.

Table 63: One-off cost. Foundation models: compute of the final pretraining run(s) as reported by their authors (quoted verbatim, with URL and page, in the supplementary material); "not reported" where the primary sources are silent. GPU-hours are computed only from stated numbers and are not comparable across GPU types. TabEvo: the final LLM-guided search run. Development compute of every method, ours included, is excluded.

Method	One-off cost	Hardware	Wall-clock	GPU-hours
TabICL v1.1 (Qu et al., 2025)	pretraining (synthetic)	3 NVIDIA A100 40GB	20 days (16 + 3 + 1 days for stages 1-3)	$\geq$1,440
TabICLv2 (Qu et al., 2026)	pretraining (synthetic)	NVIDIA H100 80GB	not reported (only GPU-days stated)	588 per checkpoint
TabPFN v2 (Hollmann et al., 2025)	pretraining (synthetic)	8 NVIDIA RTX 2080 Ti	around 2 weeks per training run	2,688 per checkpoint
TabPFN v2.5 (Grinsztajn et al., 2025)	pretraining (synthetic+real)	not reported	not reported	not reported
TabPFN v3 (Grinsztajn et al., 2026)	pretraining (synthetic)	not reported	not reported	not reported
TabFM (Kong & Das, 2026)	pretraining (synthetic)	not reported	not reported	not reported
Mitra (Zhang et al., 2025b)	pretraining (synthetic)	8 NVIDIA A100 40GB	approximately 60 hours	480
TabDPT (Ma et al., 2025)	pretraining (real)	NVIDIA A100 40GB	not reported	not reported
LimiX (Zhang et al., 2025a)	pretraining (synthetic)	not reported	not reported	not reported
TabEvo (ours)	LLM-guided search, 15 tables[a]	8 H100 (shared)	11.7 h	$\approx$46 ($\leq$86)[b]
	LLM: 407 calls, 17.8M prompt + 0.82M completion tokens (hosted API models)			

Foundation models: final pretraining run(s) as reported by the authors; GPU-hours computed only from stated numbers and not comparable across GPU types; development compute of every method is excluded. [a] The final search run only; the development of the seed architecture it started from is not included. [b] Estimate from pre-run worker-hours per child; the upper bound is 8 GPUs $\times$ the hours with an evaluation in flight.

I.8 Results by regime

J Unstructured Python-program search

Method. In this exploratory search, an LLM evolved the whole Python `fit_predict(X_ctx, y_ctx, X_qry)` function by MAP-Elites mutation; candidates could also `pretrain` and carry a checkpoint. Unlike TabEvo's real-table search, it used a diverse synthetic prior: 104 tasks from 59 families, scored by mean normalized loss over 16 task-type/size cells (cross-entropy$/\log K$ or $\log(1+\mathrm{MSE}/\mathrm{Var}(y_{\mathrm{ctx}}))$). Island A allowed in-episode library fitting; island B required pretraining, banned library predictors at inference, and required a checkpoint-dependence loss increase of 0.03. Across stages with revised settings, about 3,700 candidates consumed about 530 evaluation-hours; the last stage evaluated 823 of its 1,000 planned candidates. Table 65 shows its leading programs.

Table 64: **Regimes.** $\Delta = TabEvo-$method in points with its 95 % bootstrap interval, wins/ties/losses from TabEvo's side, and the marks of Table 1 (Holm correction within each column group), on the unseen tables with at most 4,000 training rows, those with more, and the high-dimensional tables.

	Unseen, ≤4k dev rows (16)		Unseen, >4k dev rows (17/20)		High-dim. (20)	
TabEvo vs.	Δ [95% CI]	W/T/L	Δ [95% CI]	W/T/L	Δ [95% CI]	W/T/L
TabFM	−1.16 [−1.88, −0.42]	2/1/13	−0.93 [−1.76, −0.33]	3/1/13	−2.41† [−3.40, −1.42]	2/1/16 ↑
TabPFN v3	−0.69 [−1.36, −0.02]	5/1/10	−0.67 [−1.12, −0.29]	4/0/13 ↑	−1.49† [−2.50, −0.49]	6/0/13
TabPFN v2.5	−0.49 [−1.15, 0.13]	4/3/9	−0.44 [−0.78, −0.17]	3/1/13 ≈	−1.51 [−2.64, −0.45]	6/1/13
TabPFN v2	−0.03 [−0.89, 0.89]	4/2/10 ≈	+0.04 [−0.42, 0.59]	5/2/10 ≈	−0.02 [−1.79, 1.89]	7/1/12
TabICLv2	−0.23 [−1.06, 0.63]	5/1/10	−0.42 [−0.68, −0.18]	4/0/13 ≈	−0.25 [−1.61, 1.21]	8/1/11
Stack-TD	−0.15 [−0.88, 0.47]	9/1/6 ≈	−0.18 [−0.73, 0.29]	6/1/10 ≈	–	–
CatBoost-TD$_{b5}$	+0.18 [−0.49, 0.78]	11/1/4 ≈	−0.22 [−0.80, 0.22]	5/1/11 ≈	−0.36 [−1.78, 1.04]	10/2/8
RealMLP-TD$_{b5}$	+0.18 [−0.48, 0.80]	9/2/5 ≈	+0.30 [−0.27, 1.06]	6/2/9	+0.18 [−1.88, 3.07]	9/1/10
TabM-D$_{b5}$	+0.62 [−0.14, 1.23]	10/3/3	+0.41 [−0.26, 1.40]	6/2/9	+1.17 [−1.45, 5.04]	11/0/9

Table 65: **Python-program search on the synthetic prior** (exploratory). Normalized loss on 104 tasks, lower is better; checkpoint dependence is the loss increase after replacing trained weights by a checkpoint trained for 15 seconds.

Program	Macro loss	Classification	Regression	Ckpt. dependence
Context constant	0.786	–	–	–
A: open program	**0.523**	**0.616**	**0.430**	0
B: amortized program	0.545	0.646	0.445	0.137

Transfer and comparison with TabEvo. We froze A (CatBoost–ExtraTrees–kernel ensemble) and B (pretrained feature-token embedding with kernel smoothing). Both completed all 71 real-table splits, none used in their synthetic search, with at most 768 stratified context rows and one seed. Table 66 compares them on TabEvo's completed cells; the group names follow the transfer benchmark. TabEvo fitted up to 10,000 rows, so this is a descriptive cross-protocol comparison, not an equal-budget representation test.

Table 66: **Real-table transfer** (exploratory). Accuracy (%) or R^2 on common completed tables; n/N is TabEvo coverage (A/B: all 71/71).

Group	Task	TabEvo n/N	TabEvo	A: open	B: amortized
Fresh	classification	21/26†	**88.19**	86.75	85.20
Held-out	classification	5/5	**89.43**	87.53	83.65
Held-out	regression	5/5	**0.756**	0.593	0.565
Battery	classification	7/7	**82.88**	81.10	78.62
Battery	regression	8/8	**0.810**	0.693	0.639
High-dimensional	classification	7/13	**91.35**	87.21	68.79
High-dimensional	regression	7/7	**0.583**	0.259	0.313

†Compared on 21 fresh tables; TabEvo has results on 23 of the 26 (Table 1).

TabEvo has the highest mean on every common-completion slice, while A and B complete all 71 tasks (TabEvo's failures are scored in Table 1). A beats B on 39/51 classification tables (seven ties) and 14/20 regression tables; B has higher mean R^2 on high-dimensional regression.