AGENT-DESIGNED SEARCH SPACES FOR TABULAR MACHINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite the rapid progress of LLM-based agents for planning, code generation, and debugging, their practical value for tabular machine learning remains underexplored. In this paper, we investigate a concrete use case: whether state-of-the-art agentic AI systems can design extended HPO search spaces for established tabular models that outperform the standard published search spaces of these models. Specifically, we represent each tabular model as a modular pipeline covering preprocessing, embeddings, architecture, training, and inference. We then task the agent to propose candidate code implementations for each module and use a classical HPO algorithm to jointly optimize over these candidates and the model's default hyperparameters. Compared with the base HPO spaces, the expanded search spaces improve four of the five model families across a suite of 45 datasets, with average relative gains of 0.6%, rising to 2.0% on small-to-medium regression datasets. Notably, these gains are obtained at matched budgets: the enlarged spaces outperform the base under the same number of tuning trials and ensemble members. The gains transfer to the recent TabArena benchmark, where the tuned and ensembled agentic spaces improve the Elo of four of the five model families and the two strongest agentic ensembles surpass AutoGluon's ensemble of conventional models. Under matched budgets, neither an end-to-end agent loop nor a human-curated bank of AutoML-library components reproduces these gains, suggesting that the value of LLM agents lies in expanding the design space. The code, prompts, and generated search spaces will be released upon publication.

1 INTRODUCTION

Tabular data is one of the most abundant modalities in machine learning, with applications across finance, healthcare, recommender systems, and scientific data analysis. This practical importance has made tabular ML an active research area, with recent progress in specialized DNN architectures (Holzmüller et al., 2024; Gorishniy et al., 2025; Ye et al., 2025), foundational models (Hollmann et al., 2023a; Qu et al., 2025; Zhang et al., 2025), and more rigorous benchmarks and evaluation protocols (Rubachev et al., 2025; Erickson et al., 2025).

At the same time, the recent rise of large language models (LLMs) has so far had only limited impact on state-of-the-art tabular ML. While prior work has investigated LLMs for direct prediction over serialized tables (Hegselmann et al., 2023; Gardner et al., 2024), semantic feature engineering (Hollmann et al., 2023b; Han et al., 2024), and prediction with textual metadata (Grinsztajn et al., 2023; Yakushev et al.), these approaches appear most useful in specific niches, e.g., when datasets are small or feature names are semantically informative, and have had little effect on standard practitioner workflows.

The ability of recent AI agents to analyse and improve code suits an under-automated part of tabular ML practice: *identifying which recent techniques or combinations of them can improve a particular model*. Most tabular models are released as a frozen recipe with a small hyperparameter-optimization (HPO) space. In practice, users hand-tune models from domain knowledge and literature trends (e.g. a new activation function or optimizer) with no systematic way to explore the options or their interactions. We investigate whether state-of-the-art agents can close this gap automatically: given a model, we split it into code modules (e.g. preprocessing, embedding, architecture), task the agent to propose and implement alternatives for each, add them as categorical variables to the HPO space,

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.

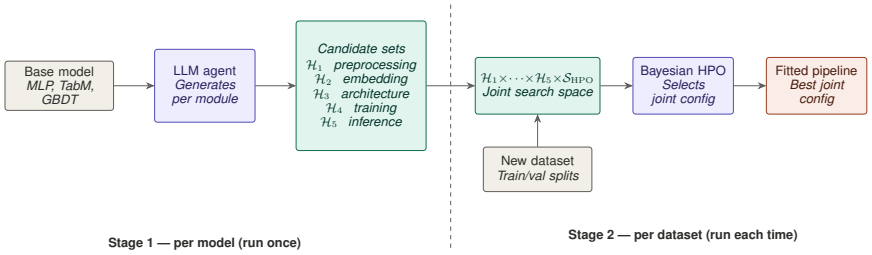


Figure 1: Method overview. *Stage 1* runs once per base model: an LLM agent generates candidate sets $\mathcal{H}_1, \ldots, \mathcal{H}_5$ for the five pipeline modules. *Stage 2* runs on each new dataset: a classical HPO searches the joint space and returns a fitted pipeline.

and tune the model over this extended space with a standard HPO algorithm (Figure 1). This use of LLMs is attractive for several reasons. First, it improves performance: for most model families, the enlarged search space lets HPO find variants that outperform the model tuned over its published HPO space. Second, it can simplify model upkeep: candidate techniques are proposed and implemented once per model rather than tested by hand. Third, the cost of the LLM can be amortized across the datasets: the search space can be generated once per-model without access to the target data.

On a curated set of 45 datasets, the agent-augmented spaces improve the aggregate rank of nearly every family over the default HPO spaces: averaged across all datasets, tuned single models gain 0.5% for DNNs (MLP†, TabM†, RealMLP), 0.8% for GBDTs (LightGBM), and 0.5% for the TabICLv2 foundation model, while ensembles gain 0.9% for DNNs and 0.2% for GBDTs. The gains concentrate on small-to-medium regression datasets, where the average single-model gains rise to 2.0% for DNN and GBDT, while TabICLv2 improves on large regression datasets by 2.1%. The richer spaces also reduce prediction correlation and thus strengthen hyperparameter ensembles. The conclusions transfer to the recent TabArena benchmark, where, with tuning and ensembling, the agentic spaces improve the Elo of four of the five model families by up to 125 points.

The contributions of our paper are the following:

1. We propose a new method for improving tabular models with LLM agents that never exposes the target data to the agent: the researcher splits the model into a sequence of modules, the agent generates candidate module implementations, and a traditional HPO algorithm explores the resulting joint search space on each dataset.
2. We release our prompts, harnesses, and generated search spaces for five model families (MLP†, TabM†, LightGBM, RealMLP, TabICLv2). The spaces can be plugged into existing AutoML systems: adding agentic RealMLP and TabM† configurations to AutoGluon improves it.
3. We verify the gains on two benchmarks. On our curated 45-dataset suite, agentic spaces improve four of five families at matched budgets, most on small-to-medium regression and ensembles, and agentic RealMLP yields the strongest single-family ensemble. On TabArena, agentic TabICLv2 and RealMLP surpass AutoGluon's `best_quality` preset.
4. We show what fails and what drives the gains. An end-to-end agent loop trails classical HPO at \$20–50 per dataset, and a human-curated bank of AutoPyTorch components in our pipelines does not reproduce the gains. No candidate is universally best: gains come from per-dataset composition and ensemble diversity and hold across two agents and ten re-generations.

2 Related Works

Tabular deep learning and pipeline design. Recent tabular work has advanced both architectures (TabM (Gorishniy et al., 2025), TabR (Gorishniy et al., 2024), ModernNCA (Ye et al., 2025)) and foundation models (TabPFN (Hollmann et al., 2023a), TabICLv2 (Qu et al., 2026), LimiX (Zhang et al., 2025)), while benchmarks like TabReD (Rubachev et al., 2025) and TabArena (Erickson et al., 2025) show that protocol choices substantially affect conclusions.

AutoML and architecture search. Auto-sklearn (Feurer et al., 2015) builds a greedy ensemble (Caruana et al., 2004) over combinations of preprocessing methods and shallow models. AutoGluon (Erick-

son et al., 2020) follows the same ensemble-first recipe, but does not optimize over preprocessing and broadens the model library from shallow learners to deep architectures (e.g. TabM (Gorishniy et al., 2025), RealMLP (Holzmüller et al., 2024)) and foundation models (e.g. TabICLv2 (Qu et al., 2026), LimiX (Zhang et al., 2025)). AutoPyTorch (Zimmer et al., 2021) instead ensembles parametrizations of a single model family, varying regularizers, activations, and MLP layer shapes. AutoKeras (Jin et al., 2019) similarly searches over neural architectures assembled from its own library of implemented components (Zoph & Le, 2017; Liu et al., 2019). Our setting is closest to AutoPyTorch and AutoKeras: we start from a single model and search over its modifications. We differ in two key ways: the candidates are generated conditioned on the input model, i.e., proposed specifically for it, and they are written by the agent rather than hand-implemented. The resulting models with extended HPO spaces can then be wrapped and used with Auto-sklearn and AutoGluon.

LLMs for tabular prediction. Prior LLM-based approaches to tabular prediction fall into two families. The first treats the LLM as a direct predictor over serialized rows, via prompting (TabLLM (Hegselmann et al., 2023)) or fine-tuning (Tabula-8B (Gardner et al., 2024)); the second uses the LLM for semantic feature engineering, generating new columns from dataset descriptions (CAAFE (Hollmann et al., 2023b), FeatLLM (Han et al., 2024)). Both have shown gains in regimes such as small datasets or rich textual metadata.

LLM agents for optimization and research automation. Prior work uses LLMs either as optimizers (Liu et al., 2024; Chen et al., 2022; Meindl et al., 2025; Menet et al., 2025; Tan et al., 2025) or as agents that automate parts of the research loop (Lu et al., 2024; Karpathy, 2026; Ferreira et al., 2026). The former is orthogonal to our work: these methods change how a fixed search space is explored, whereas we use a standard sampler and change the space itself. Our work is closer to the latter, but restricts the agent to generating modules along predefined pipeline axes, followed by controlled HPO. This preserves dataset privacy, amortizes agent cost, and turns the search space into a fixed, inspectable artifact that can be released and ablated.

3 METHODOLOGY

3.1 SETUP AND NOTATION

Datasets. We consider a tabular task with a fixed split $\mathcal{D} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}})$. We use datasets derived from TabM (Gorishniy et al., 2025), TabArena (Erickson et al., 2025), and TabReD (Rubachev et al., 2025) benchmarks, which span both regression and classification datasets ranging from 768 to 1M+ objects and from 5 to 1500+ features (summary statistics in Table 6, full properties in Appendix B.2). Our design choices for the benchmark were to: (1) include both i.i.d. and non-i.i.d. datasets; (2) keep a roughly balanced split between regression and classification; (3) include datasets with a diverse range of sample-to-feature ratios; we separately report performance on the TabArena benchmark in Appendix A.

Metrics. We evaluate every dataset with its native metric adopted from the source benchmark (AUROC, accuracy, or log-loss for classification and RMSE for regression); the per-dataset metrics are listed in Appendix B.2. We report four aggregate metrics, all at a fixed tuning budget B: mean rank, relative improvement over the tuned base-space MLP†, normalized score, and Elo score (defined in Appendix B.3).

Models and agents. We select strong within-family implementations. To represent DNNs, we take, in order of increasing complexity: (i) MLP† – an MLP with piecewise-linear embeddings of Gorishniy et al. (2022); (ii) TabM† – the mini variant of TabM with piecewise-linear embeddings and $k = 32$ members from Gorishniy et al. (2025); (iii) RealMLP – the MLP model from PyTabKit with the latest TabArena hyperparameters (Holzmüller et al., 2024; Erickson et al., 2025). To represent GBDT, we take the LightGBM model (Ke et al., 2017). Among foundational models, we take the recently proposed TabICLv2 (Qu et al., 2026); since it performs no gradient-based training, we refit it on the concatenated train and validation splits to obtain test predictions (Appendix B.4). As an agent, we use Claude Code running Opus 4.8 on `max` effort settings (Anthropic, 2026a;b). We additionally run speed and performance ablations with Codex running GPT 5.5 with `extra-high` settings (OpenAI, 2025; 2026a).

Evaluation setup. We tune each model over its HPO space with Optuna's univariate Tree-structured Parzen Estimator (TPE) sampler (Bergstra et al., 2011; Akiba et al., 2019). The first 20 trials are sampled at random to seed the densities. Each trial trains one configuration on the training subset and scores it on the validation subset; the budget is 200 trials on small and medium datasets and 100 on the large ones (Microsoft and TabReD), after which the configuration with the best validation score is evaluated on the held-out test subset with 15 random seeds. TabICLv2 is the exception: the default model is used as published, without tuning, while its agentic variant is tuned with a fixed budget of 100 trials on every dataset. For the ensembling study, we build greedy ensembles following Caruana et al. (2004): we sample 100 random configurations per search space (20 on large datasets), train them, and iteratively add, with replacement, the model whose inclusion in the prediction average most reduces the validation error. The selected ensemble is then evaluated on the test subset; further details on the greedy ensemble algorithm are given in Appendix B.5.

3.2 From end-to-end agents to agent-generated search spaces

The most direct way to hand a tabular model to an agent is to let it edit the model end to end: starting from the reference implementation, the agent changes a hyperparameter or a piece of code, runs a validation fit, and keeps the change if the score improves. This is the autoresearch loop of Karpathy (2026), which we adapt to the tabular setting and run for 50 iterations (one run per dataset) on four datasets, starting from the $\mathrm{MLP}^\dagger$ code (Appendix D.4). Table 1 summarizes the outcome: the loop costs \$20–50 *per dataset*, requires handing the agent the data, and ends up marginally behind classical HPO over the default HPO space.

Table 1: The end-to-end autoresearch loop (Karpathy, 2026) against classical HPO and our method, on the four datasets of Table 15 at a matched budget of 50 trials. Δ is the mean relative improvement.

Method	Agent is run	Agent cost	Sees the data	Δ over classical HPO
Autoresearch (Karpathy, 2026)	per dataset	\$20–50 per dataset	yes	-0.1%
Ours	per model	\$10 per model	no	$+1.3\%$

Three problems. We attribute this negative result to three properties of the end-to-end loop.

(i) Few ideas generalize well in tabular data; diversity and number are therefore an advantage. Within the most important modules, no candidate wins on more than 12 of the 45 datasets and every candidate wins on at least one (Section 5.1), and performance grows with the size of the candidate pool (Section 5.3). No single module generalizes across the datasets, so the search should maximize the *diversity* and *number* of explored ideas. The end-to-end loop instead tends to converge on a narrow set of techniques: current LLMs tend to explore poorly in-context (Krishnamurthy et al., 2024), and agents in long loops increasingly revisit earlier actions as their context fills up (Wang et al., 2026; Xu et al., 2026).

(ii) The search space is explored inefficiently. Even for the ideas it does try, the greedy LLM-based search evaluates one edit at a time, accepts greedily, keeps no model of the response surface, and proposes candidates purely from the LLM prior. The weakness is sharpest for the numerical hyperparameters, which hold a large share of the achievable gain (learning rate, weight decay; Figure 3, blue boxes). On exactly such problems, classical black-box optimizers such as TPE outperform LLMs (Ferreira et al., 2026).

(iii) The cost does not amortize across datasets. The loop is re-run for every new dataset: repeating our 45-dataset study with 50 trials per dataset would cost at least \$900–\$2,300 in agent calls alone. The agent also needs the data to score its own edits, which invites contamination.

Three fixes. Our method addresses each problem in turn.

(i) Steer the agent towards coverage. We decompose each model by hand into a diverse set of mutually non-overlapping modules supported by the literature. For the neural families ($\mathrm{MLP}^\dagger$, $\mathrm{TabM}^\dagger$, and RealMLP), we take the modules to be preprocessing, feature embeddings, architecture, training, and inference with the default taken from the original method, e.g., for $\mathrm{MLP}^\dagger$ quantile-transform as numerical preprocessing, AdamW (Loshchilov & Hutter, 2019) as optimizer. For LightGBM and TabICLv2, whose learning algorithms are frozen, we take the modules to be preprocessing

and inference. The agent proposes alternative implementations of each module as self-contained code blocks, e.g., Yeo–Johnson feature transform (Yeo & Johnson, 2000), Lion optimizer (Chen et al., 2023). We instruct the agent to generate at most 64 candidates per task type (regression and classification), asking it to filter out proposals that are repetitive, do not pass the evidence bar (i.e., already tried and failed in the literature), or repeatedly fail the synthetic smoke tests (Appendix D.3). Each module then enters the search space as a categorical variable with a uniform prior, e.g., train module $\sim \mathrm{Categorical}(\{\text{AdamW}, \text{AdamW + EMA}, \dots\})$. Our design promotes coverage of promising configurations.

(ii) Delegate the exploration to a classical sampler. The agent only generates module alternatives, expressed as code. The joint space of modules and continuous hyperparameters is then explored by a standard HPO sampler, dividing the work between the agent and the HPO algorithm.

(iii) Generate once per model family. Concretely, the agent receives the model's code, the compressed title and abstract of the original paper, and tools to test candidate implementations, but no target dataset. The generation cost then amortizes over every later dataset, and leakage of the target data into the agent is excluded by construction.

In Appendix D.3, we show full sets of generated modules for each model and the complete agent environment.

Alternatives. The design above deliberately produces a single space per model, which standard HPO then adapts per dataset. One could instead generate a space per dataset or admit conditional hyperparameters; we avoid both for the reasons in problem (iii) and because public benchmarks are mostly small to medium, so an exponentially larger conditional space would likely overfit, although tailoring the space to a private large dataset may prove beneficial.

4 Results

4.1 Main performance

Tuned models. Figure 2 (top) summarizes the tuned single-model results: in four of the five model families, the agent-augmented space improves the aggregate standing over its base space, consistently across all four metrics; the exception is RealMLP, whose base space is meta-tuned. Table 2 breaks the effect down by task type and dataset size (size groups defined in Appendix B.2). The gains are cleanest on small-to-medium regression datasets: agentic MLP† improves over its base space by $+3.7\%$ on small and $+1.6\%$ on medium datasets, agentic LightGBM by $+2.1\%$ and $+1.8\%$, and agentic TabM† by $+1.8\%$ and $+1.6\%$; on large regression datasets the improvements shrink to -0.4–$+0.1\%$. Classification is more saturated: agentic MLP† still gains $+2.1\%$ on small data, while on medium datasets the DNNs and LightGBM move within $\pm 0.4\%$ and on large ones the gains stay below $+0.7\%$. TabICLv2 is already the strongest single model on small datasets, leaving little room for improvement ($+0.1$–$+0.4\%$), but benefits from the agentic space exactly where it lags behind: $+2.1\%$ on large regression and $+1.1\%$ on large classification. RealMLP shows the saturation effect in its strongest form: its base space ships with meta-learned defaults, and the agentic space leaves the tuned single model essentially unchanged on regression while degrading classification by -0.8% on medium and large datasets. The main conclusion is therefore not that an agent always helps, but that agent-generated spaces help most when the base family exposes useful, composable modules and the task is not already saturated.

Tuned and ensembled models. In Figure 2 (bottom), ensembling amplifies the effect: ensembles drawn from the agentic spaces occupy the top of the leaderboard, and the agentic RealMLP ensemble is the strongest single-family ensemble, behind only the multi-model AutoGluon system (per-dataset tables in Appendix E.3). The ensemble rows of Table 2 follow the single-model pattern but are more uniform across task groups: agentic MLP† gains $+3.2\%$ on small and $+1.6\%$ on medium regression and $+2.1\%$ on small classification, agentic TabM† reaches $+2.9\%$ on small classification, and RealMLP gains $+0.9\%$ on small and $+1.0\%$ on large regression. Table 3 explains where these gains come from: the best single member of an agentic pool stays within -1.0 to $+0.4\%$ of its base counterpart, but its predictions are substantially less correlated, so the agentic space helps ensembles more than single models.

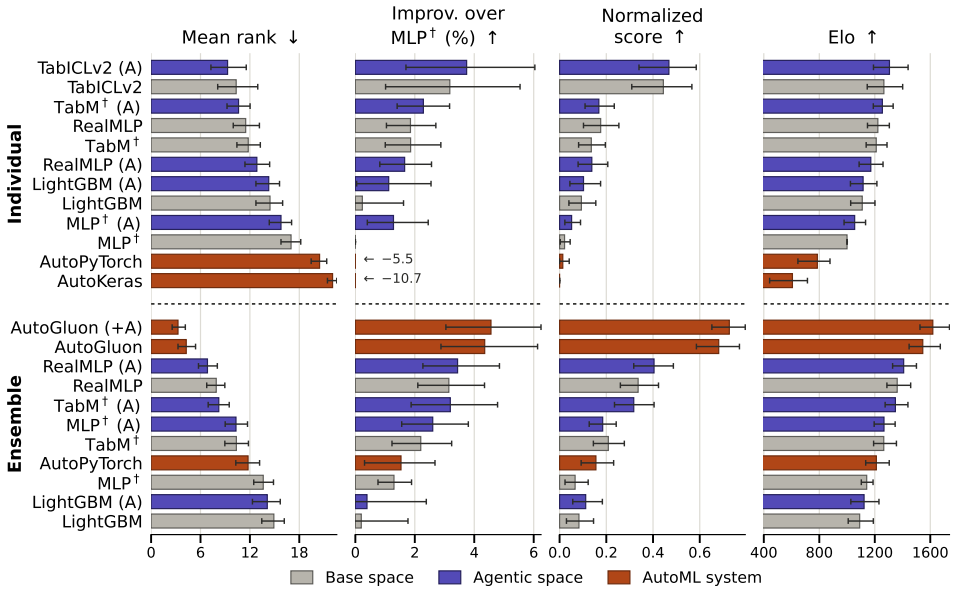


Figure 2: Aggregate metrics across 45 datasets for tuned individual models (top) and greedy ensembles (bottom) over the joint pool of all shown methods under the budgets of Section 3.1. Improvement over MLP is relative to the tuned base-space MLP† on the dataset-native metrics (axis truncated at zero, off-scale values annotated). Purple bars and (A): agent-augmented spaces; orange: AutoML systems run as-is; AutoGluon (+A): AutoGluon with the agentic RealMLP and TabM† spaces. Error bars: 95% CIs across datasets.

Table 2: Mean relative improvement (%) of the agent-augmented over the base HP space on the dataset-native metrics, for tuned single models and greedy ensembles by task type and dataset size (small/medium/large: regression 8/6/6, classification 10/12/3 datasets). TabICLv2 is not in the ensembling study.

		Regression			Classification		
Method	Scope	Small	Medium	Large	Small	Medium	Large
MLP†	Single	+3.7	+1.6	+0.1	+2.1	−0.4	+0.5
	Ensemble	+3.2	+1.6	−0.6	+2.1	+0.5	+0.2
TabM†	Single	+1.8	+1.6	−0.1	−0.2	−0.4	+0.7
	Ensemble	+1.5	+1.5	−0.4	+2.9	0.0	+0.6
LightGBM	Single	+2.1	+1.8	−0.4	+1.2	−0.1	+0.2
	Ensemble	+1.0	+2.6	−2.3	+1.0	−0.8	−0.4
TabICLv2	Single	+0.1	+0.2	+2.1	+0.4	+0.1	+1.1
	Ensemble	–	–	–	–	–	–
RealMLP	Single	+0.3	+0.3	−0.3	0.0	−0.8	−0.8
	Ensemble	+0.9	−0.6	+1.0	+0.4	+0.1	0.0

AutoML systems. Figure 2 also compares the search spaces against three AutoML systems run as-is on the same datasets and splits: AutoKeras (Jin et al., 2019), AutoPyTorch (Zimmer et al., 2021), and AutoGluon (Erickson et al., 2020). Following Gijsbers et al. (2024) and Erickson et al. (2025), we set the budget to 4 hours per dataset (Appendix D.1). AutoKeras and AutoPyTorch search over libraries of human-implemented neural components and are the natural baselines for our agentic MLP†. They lag behind it in both the single-model and the ensemble regime. One factor could be that their libraries leave most of the pipeline fixed (no numerical embedding, target preprocessing, or inference modules): their 44 and 4 interchangeable components sit mostly on preprocessing and the optimizer, against 57–61 spread over all axes for the agentic MLP† space (Appendix D.1). We also test porting a human-curated bank of AutoPyTorch components into the MLP†, TabM†, and RealMLP

Table 3: *Left:* greedy-ensemble diversity over all 45 datasets: mean pairwise test-prediction correlation ρ among selected members (lower $\rightarrow$ more diverse) and mean relative improvement Δ_{base} of the agentic over the base space for the pool's best member and its ensemble. *Right:* agent output tokens and generation time (HH:MM) per model family (one generation per family).

		MLP†	TabM†	LightGBM	RealMLP
Member corr. $\rho \downarrow$	Base	0.964	0.981	0.913	0.974
	Agentic	0.892	0.872	0.799	0.957
Δ_{base} (%) $\uparrow$	Best member	+0.1	+0.4	−1.0	0.0
	Ensemble	+1.3	+1.1	+0.2	+0.3

Method	Output tokens		Time	
	Claude	Codex	Claude	Codex
MLP†	80,965	92,254	01:34	00:55
RealMLP	90,579	102,542	01:24	01:11
TabM†	64,142	71,865	01:40	01:04
LightGBM	46,221	36,961	00:43	00:34
TabICLv2	36,221	32,456	00:30	00:24

pipelines under the identical protocol, which holds the HPO and ensemble logic fixed. We report the results in Appendix D.2 and find that the ported set does not reproduce the agentic gains: the tuned MLP† and TabM† change by -1.6% and -0.6% versus $+1.7\%$ and $+0.6\%$ with the agentic spaces, and head to head the agentic set wins on 62–74% of the datasets. For AutoGluon we use the `best_quality` preset and pass it both the train and validation sets. AutoGluon fits a meta-learned portfolio of 110 fixed configurations across seven conventional model families (Appendix D.1), 8-fold bags each one, stacks the out-of-fold predictions, and greedily weights the final outputs. This design is complementary to ours, and we test the combination directly: AutoGluon (+A) receives 8 randomly-sampled configurations of our agentic RealMLP and TabM† models. The augmented system outperforms the base, selecting the agent-implemented models into its final weighted ensemble on 43 of the 45 datasets with 54% of the ensemble weight on average.

TabArena. Finally, the conclusions transfer to the recent TabArena benchmark (51 datasets, official protocol; Appendix A). With tuning and ensembling, the agentic spaces improve the Elo of MLP†, TabM, RealMLP, and TabICLv2, placing both the agentic TabICLv2 and the agentic RealMLP above AutoGluon with the `best_quality` portfolio of conventional models (the `extreme_quality` preset builds an ensemble over foundation models and stays ahead). The agentic RealMLP thus overtakes AutoGluon on TabArena but not in Figure 2, and the likely reason is the protocol. Under TabArena, every configuration is itself an 8-fold cross-validated ensemble, i.e., AutoGluon's own bagging machinery, and the greedy ensemble draws on all 200 randomly sampled tuning configurations. In our benchmark, each configuration is fit once on the training subset and scored once on the validation subset, and the greedy ensemble draws on 100 fresh random configurations (20 on large datasets), because TPE trials are highly correlated and cannot be reused. Both choices are deliberate: TPE strengthens the individual models, and forgoing cross-validation admits larger datasets and grouped or temporal splits, for which cross-validation is not well defined, so Figure 2 gives a more conservative view of the individual model families.

4.2 Agent cost and amortization

Table 3 reports output tokens (thinking and generation) and wall-clock generation time for Claude Code Opus 4.8 and Codex GPT 5.5; input and cached tokens are omitted, because they are dominated by the agent's fixed system prompt and tool setup. At current per-token prices (Anthropic, 2026c; OpenAI, 2026b), one generation loop costs approximately 10 USD. The agent runs once per model family, while HPO runs per dataset: generation takes roughly 35–100 minutes depending on model family and agent, but unlike autoresearch (Karpathy, 2026) the cost is amortized across all later datasets.

5 Analysis

5.1 Which modules matter?

Figure 3 (left) shows that fANOVA assigns substantial importance to several modules extended by the agent: the architecture, the categorical and numerical embeddings, the numerical preprocessing, and the training module rank directly after the network depth. Figure 3 (right) shows that no candidate

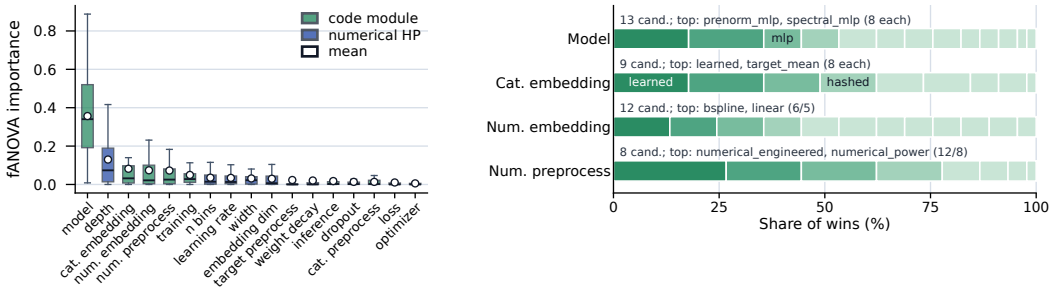


Figure 3: *Left:* fANOVA module importance for MLP†, normalized per dataset (green: agent-implemented code modules; blue: numerical hyperparameters). *Right:* share of the 45 best-HPO-trial wins per candidate of the four most important modules, sorted by share; the two most selected candidates and their win counts are named above each bar.

Table 4: Stability of agent-generated MLP† search spaces over five re-generations per agent on all 45 datasets, in the joint pool of the 12 spaces (base, released, ten re-generations; Elo anchored at the base MLP† = 1000). Agent rows: mean $\pm$ 95% t-interval half-width over the five runs.

Search space	Mean rank ↓	Norm. score ↑	Elo ↑	Improv. over MLP† (%) ↑
MLP† (base space)	8.49	0.13	1000	0.0
Agentic MLP† (released)	6.96	0.25	1095	+1.28
Claude re-generations	6.33 ± 1.03	0.25 ± 0.09	1132 ± 61	$+0.78 \pm 0.63$
Codex re-generations	6.18 ± 0.36	0.26 ± 0.06	1141 ± 22	$+0.97 \pm 0.42$

dominates within these modules: every candidate wins the best trial on at least one dataset, and the most selected candidate of a module wins on at most 12 of the 45 datasets (the engineered numerical preprocessing), with the top candidates of the other three modules winning on 6–8.

Aggregating the winning HPO trials across the 18 MLP regression datasets, every candidate except two training-side ones appears in at least one winning combination, and no two datasets select the same combination. Which candidates win is thus largely dataset-specific: the agent contributes a reusable basis of implementations, and the per-dataset composition has to be found by HPO, consistent with the candidate-pool ablation of Section 5.3.

5.2 Agent choice and stability

Table 4 compares agent-generated MLP† search spaces across five independent re-generations per agent (Claude Code with Opus 4.8 at `max` and Codex with GPT-5.5 at `xhigh` thinking level) on all 45 datasets. Every re-generated space improves over the base MLP† space on all four aggregate metrics, and the released space lies within the re-generation spread (below its mean Elo), so the main results do not rest on a favourable draw. The two agents perform on par: Codex is somewhat more consistent (Elo 1141 ± 22 vs. 1132 ± 61), while the best and the worst single runs both come from Claude (Elo 1182 and 1056). Per-run results are in Table 21.

5.3 Candidate-pool size

The spread of winning candidates in Section 5.1 suggests that growing the search space should help, but a larger space could also exacerbate overfitting to the validation subset. We check both by re-fitting the agentic MLP† over progressively subsampled candidate pools (Table 5). All four aggregate metrics improve monotonically in the mean with the pool size, and the full pool is the best on every metric (Elo 1143 vs. 1086 for the largest subsets), with no downturn at these pool sizes, so growing the space helps and validation overfitting does not yet set in at this scale.

Table 5: Candidate-pool subsampling for the agentic MLP† space on 18 random datasets: five random subsets of the full pool (61 regression / 57 classification candidates) per size in $\{8, 16, 32, 48\}$, each with the full 200-trial HPO; last row: complete pool. Metrics are computed per replicate in a pool of six spaces (base, one subset per size, full; Elo anchored at the base MLP† = 1000); mean $\pm$ 95% t-interval half-width over the five replicates.

# Candidates	Mean rank ↓	Norm. score ↑	Elo ↑	Improv. over MLP† (%) ↑
0 (base)	3.84 ± 0.11	0.18 ± 0.05	1000 (anchor)	0.0
8	3.52 ± 0.42	0.21 ± 0.10	1015 ± 53	$+0.36 \pm 0.71$
16	3.40 ± 0.30	0.24 ± 0.14	1026 ± 40	$+0.49 \pm 0.59$
32	3.33 ± 0.55	0.31 ± 0.14	1051 ± 71	$+0.57 \pm 1.00$
48	2.94 ± 0.82	0.35 ± 0.20	1086 ± 105	$+0.73 \pm 0.56$
Full	2.68 ± 0.18	0.49 ± 0.04	1143 ± 11	$+1.15$

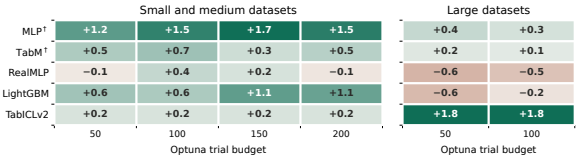


Figure 4: Mean relative improvement (%) of the agentic over the base space at matched Optuna budgets on the dataset-native metrics for small-and-medium (left) and large (right) datasets; green: agentic space ahead. TabICLv2 cells are constant (default untuned; agentic tuned at 100 trials).

5.4 Trial budget.

Since larger spaces need more trials, Figure 4 reports the average improvement of the agentic over the base space at each matched budget. On small-and-medium datasets the gap is positive at every budget for MLP†, TabM†, and LightGBM, while the agentic RealMLP stays within $\pm 0.4\%$ of its base; on large datasets the gaps are smaller and mixed, except for TabICLv2 ($+1.8\%$). The gaps change little from 50 trials onwards, so the conclusions of Section 4.1 do not hinge on the budget within the tested range.

6 Limitations

The method expands a finite-budget HPO problem: $\mathcal{S}_{\text{HPO}} \subseteq \mathcal{S}_{\text{agent}}$ does not guarantee finite-budget improvement, larger spaces can overfit validation, and per-fit cost can rise with costlier optimizers or test-time augmentations (mean fit-time ratio to the base space from $0.9\times$ for RealMLP to $2.4\times$ for TabICLv2). Agent-generated code must be inspected for bugs before shipping, and agent products, prompts, and corpora change over time and must be logged. Single-model gains are smaller in absolute terms; the larger effects appear in ensembles and in the TabArena Elo, where pool diversity is exploited. Our evaluation spans 5 families and 2 benchmarks but does not establish universal gains. Finally, we avoid dataset-specific prompting, which protects against leakage and amortizes cost but can miss modules suited to a dataset's semantics or feature types; future directions are given in Appendix C.

7 Conclusion

LLM agents can generate executable module implementations that expand the HPO search space. Paired with classical HPO, this basis yields gains for classic DNNs, largest on small-to-medium regression and for ensembles, and smaller ones for GBDT, foundation, and meta-tuned models. The value comes from dataset-specific composition: no candidate is universally best, but the basis gives HPO more ways to adapt the pipeline, a middle ground between fixed grids and autonomous loops. The added diversity also boosts ensembles: the agentic RealMLP ensemble is the strongest single-family ensemble, behind only the multi-model AutoGluon.

REPRODUCIBILITY STATEMENT

The code repository accompanying this paper, to be released upon publication, contains the code, the prompts, the agent harnesses and tools, and the module registry with every generated module implementation for the five model families. The agent workflow, its guardrails, and the synthetic smoke tests are described in Appendix D.3, the procedure used to sanitize the agent environment between runs in Appendix D, and the complete candidate sets in Tables 16–20. The agents, model versions, effort settings, and tuning and ensembling budgets are stated in Section 3.1. Datasets, their source benchmarks, size groups, and native metrics are documented in Appendix B.2, the aggregate metrics in Appendix B.3, the base HPO space of every family in Appendix B.4, the TPE and greedy-ensembling protocol in Appendix B.5, and the hardware in Appendix B.1. The AutoML baselines and the AutoGluon portfolios are specified in Appendix D.1, the human-curated module bank in Appendix D.2, the autoresearch baseline in Appendix D.4, and the TabArena protocol in Appendix A. Per-dataset results behind every figure and table of the main text are reported in Appendices E and F, and the per-run results of the agent re-generation study in Appendix E.1.

AI USE STATEMENT

LLM agents are the object of study in this work rather than only a writing aid, so we disclose their use in both roles. As part of the proposed method, the agents named in Section 3.1 (Claude Code and Codex) proposed the candidate hypotheses and implemented the corresponding modules for each model family; an agent also drove the end-to-end autoresearch baseline of Appendix D.4, editing code and hyperparameters on the four datasets used there. Every generated module passed the synthetic smoke tests of Appendix D.3 and was inspected by the authors, the full candidate sets are released and listed in Tables 16–20, and the agent environment was sanitized between runs (Appendix D). The search-space agent never had access to any evaluation dataset; the autoresearch baseline, by design, saw the training and validation splits of its four datasets but never the test split. Outside the studied method, we did not use generative AI to develop theoretical models, formulate or prove mathematical claims, generate or clean datasets, or interpret results. We used generative AI tools to draft and edit parts of the manuscript text for readability and concision; all claims and numbers were written and verified by the authors. We have reviewed all AI-assisted text and artifacts and take responsibility for the final content of this work.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2623–2631, 2019. doi: 10.1145/3292500.3330701.

Anthropic. Claude code. `https://www.anthropic.com/product/claude-code`, 2026a. Accessed: 2026-05-07.

Anthropic. Claude Opus 4.8. `https://www.anthropic.com/news/claude-opus-4-8`, May 2026b. Large language model. Accessed 2026-06-26.

Anthropic. Claude API Pricing. `https://docs.anthropic.com/en/docs/about-claude/pricing`, 2026c. Accessed: 2026-05-06. Lists Claude Opus 4.7 pricing as $5 per million input tokens and $25 per million output tokens.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, 2011.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the International Conference on Machine Learning*, 2004. URL `https://www.cs.cornell.edu/~alexn/papers/shotgun.icml04.revised.rev2.pdf`.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Qiuyi Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems*, 2022.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the International Conference on Machine Learning*, pp. 933–941, 2017. URL https://proceedings.mlr.press/v70/dauphin17a.html.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1): 1–26, 1979. doi: 10.1214/aos/1176344552.

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.

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 (NeurIPS), Datasets and Benchmarks Track*, 2025.

Fabio Ferreira, Lucca Wobbe, Arjun Krishnakumar, Frank Hutter, and Arber Zela. Can llms beat classical hyperparameter optimization algorithms? a study on autoresearch. *arXiv preprint arXiv:2603.24647*, 2026.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1050–1059, 2016. URL https://proceedings.mlr.press/v48/gal16.html.

Josh Gardner, Juan C. Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Pieter Gijsbers, Marcos L. P. Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. AMLB: an AutoML benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 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 (ICLR)*, 2024.

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

Yury Gorishniy, Ivan Rubachev, Dmitrii Feoktistov, and Artem Babenko. Benchmarking optimizers for mlps in tabular deep learning. *arXiv preprint arXiv:2604.15297*, 2026.

Leo Grinsztajn, Myung Jun Kim, Edouard Oyallon, and Gael Varoquaux. Modeling string entries for tabular data prediction: do we need big large language models? In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

Cheng Guo and Felix Berkhahn. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*, 2016. URL https://arxiv.org/abs/1604.06737.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning*, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Sungwon Han, Jinsung Yoon, Sercan Ö. Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

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 (ICLR)*, 2023a.

Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

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*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35 (1):73–101, 1964. doi: 10.1214/aoms/1177703732.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Conference on Uncertainty in Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1803.05407.

Haifeng Jin, Qingquan Song, and Xia Hu. Auto-Keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1946–1956, 2019.

Andrej Karpathy. autoresearch: Ai agents running research loops. https://github.com/karpathy/autoresearch, 2026.

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, pp. 3146–3154, 2017.

Akshay Krishnamurthy, Keegan Harris, Dylan J Foster, Cyril Zhang, and Aleksandrs Slivkins. Can large language models explore in-context? In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 2017. URL https://arxiv.org/abs/1612.01474.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2980–2988, 2017. URL https://openaccess.thecvf.com/content_iccv_2017/html/Lin_Focal_Loss_for_ICCV_2017_paper.html.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. In *International Conference on Learning Representations*, 2024.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery, 2024.

James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297, 1967.

Jamison Meindl, Yunsheng Tian, Tony Cui, Veronika Thost, Zhang-Wei Hong, Jie Chen, Wojciech Matusik, and Mina Konaković Luković. GPTOpt: Towards efficient LLM-based black-box optimization. *arXiv preprint arXiv:2510.25404*, 2025.

Nicolas Menet, Aleksandar Terzić, Michael Hersche, Andreas Krause, and Abbas Rahimi. Thompson sampling via fine-tuning of LLMs. *arXiv preprint arXiv:2510.13328*, 2025.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001. doi: 10.1145/507533.507538.

OpenAI. Introducing Codex. https://openai.com/index/introducing-codex/, May 2025. AI coding agent. Accessed 2026-06-26.

OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, April 2026a. Large language model. Accessed 2026-06-26.

OpenAI. API Pricing. https://openai.com/api/pricing/, 2026b. Accessed: 2026-05-06. Lists GPT-5.5 pricing as $5 per million input tokens, $0.50 per million cached input tokens, and $30 per million output tokens.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pp. 1310–1318, 2013. URL https://proceedings.mlr.press/v28/pascanu13.html.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11671.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992. doi: 10.1137/0330046.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From sparse to soft mixtures of experts. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2308.00951.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning (ICML)*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026. doi: 10.48550/arXiv.2602.11139. URL https://arxiv.org/abs/2602.11139.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007. URL https://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.

Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. TabReD: Analyzing pitfalls and filling the gaps in tabular deep learning benchmarks. In *International Conference on Learning Representations (ICLR)*, 2025.

David Salinas and Nick Erickson. Tabrepo: A large scale repository of tabular model evaluations and its automl applications. *AutoML Conference (ABCD Track)*, 2024.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

Rong-Xi Tan, Ming Chen, Ke Xue, Yao Wang, Yaoyuan Wang, Sheng Fu, and Chao Qian. Towards universal offline black-box optimization via learning language model embeddings. In *International Conference on Machine Learning*, 2025.

Jerry Tang. TPE – how many startup trials are enough? Kaggle Notebook. URL https://www.kaggle.com/code/jerrytang88/tpe-how-many-startup-trials-are-enough. Accessed: 2026-07-09.

Yanbo Wang, Jinhua Hao, Yuze Shi, Kun Yuan, and Ming Sun. No time like the present: Agentic test-time training for llm agents. *arXiv preprint arXiv:2607.03441*, 2026.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1113–1120, 2009. doi: 10.1145/1553374.1553516.

Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: An alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Sklf1yrYDr.

Dongjie Xu, Hao Wu, Weijie Shi, Yue Cui, Yuanjun Liu, Jiawei Li, Haolun Ma, An Liu, Jia Zhu, and Jiajie Xu. Loopguard: Breaking self-reinforcing attention loops via dynamic kv cache intervention. *arXiv preprint arXiv:2604.10044*, 2026.

George Yakushev, Alina Shutova, Ivan Rubachev, Renat Sergazinov, and Artem Babenko. Talking trees: Reasoning-assisted induction of decision trees for tabular data. In *EurIPS 2025 Workshop: AI for Tabular Data*.

Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later. In *International Conference on Learning Representations (ICLR)*, 2025. ModernNCA.

In-Kwon Yeo and Richard A. Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000. doi: 10.1093/biomet/87.4.954.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019. URL https://arxiv.org/abs/1905.04899.

Zhang et al. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1910.07467.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, 2019. URL https://papers.nips.cc/paper/2019/hash/90fd4f88f588ae64038134f1eeaa023f-Abstract.html.

Lucas Zimmer, Marius Lindauer, and Frank Hutter. Auto-PyTorch: Multi-fidelity metalearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43 (9):3079–3090, 2021.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.

A Additional Results on TabArena

Setup. TabArena (Erickson et al., 2025) evaluates each model family under a fixed protocol: one default configuration plus 200 random configurations drawn from the family's search space, each fitted as an 8-fold bagged ensemble. *Tuned* (T) reports the configuration with the best validation score, and *Tuned+Ensembled* (T+E) reports the greedy weighted ensemble (Caruana et al., 2004) built from the same 200 configurations. We plug the agent-generated search spaces of Section 3.2 into this protocol unchanged for the five families: LightGBM, MLP†, RealMLP, TabM, and TabICLv2. We mark the agentic version with a suffix `-A`. For the four families other than MLP† the base numbers are the official TabArena entries; TabArena has no MLP entry, so we additionally run the base MLP† ourselves under the identical protocol. Since the agentic design changes the tuning space and not the default configuration, MLP† and MLP†-A share their default run, and only their Tuned and T+E results differ. Three differences from the setup of Section 4 matter when comparing the numbers. First, TabArena tunes by random search over the 200 configurations, whereas our benchmark obtains the tuned results with Optuna TPE, which is a stronger optimizer. Second, every TabArena configuration is itself an 8-fold cross-validated ensemble: it trains eight models, each on seven of the eight folds, scores the configuration on their out-of-fold predictions, and averages the test predictions of all eight models. Our benchmark trains each configuration once and scores it on the validation subset. Third, we evaluate on TabArena-Lite, i.e., the first of the seven outer cross-validation splits.

Results. Figure 5 shows the official Elo leaderboard over the 51 TabArena-Lite datasets with the agentic variants included. The picture from Section 4.1 transfers: at the T+E level, the agentic space improves the Elo of MLP† (from 1295 to 1421), TabM (from 1414 to 1497), RealMLP (from 1502 to 1557), and TabICLv2 (from 1573 for the default-only official entry to 1612), and roughly matches it for LightGBM (from 1402 to 1409). The ordering across families also matches Section 4.1: the plain MLP†, whose base space is the least tuned, gains the most ($+125$), while the meta-tuned RealMLP and the saturated LightGBM gain the least. Notably, both TabICLv2-A and RealMLP-A at the T+E level outperform the `best_quality` preset of AutoGluon 1.4 (1535)[1], and TabICLv2-A is behind only the `extreme_quality` preset of AutoGluon 1.5 (1669)[2] and newer TFMs. At the tuned-only level the gains are smaller ($+79$ for MLP†, $+14$ for RealMLP, $+3$ for TabM) or negative (-53 for LightGBM): random search is less sample-efficient in the larger agentic spaces, so, as in Section 4.1, most of the benefit is realized when the diversity of the pool is exploited by ensembling. We note that Elo is fitted over the whole method pool, so all ratings shift slightly whenever the pool changes; the numbers above are therefore comparable within this leaderboard but not against other versions of it. Per-dataset Default/Tuned/T+E scores for all five families are reported in Appendix F.

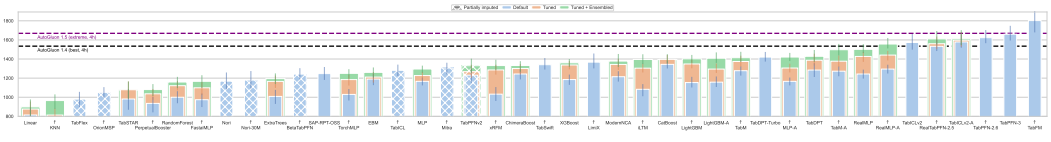


Figure 5: Official TabArena-Lite Elo leaderboard (51 datasets) with the agent-augmented spaces included as LightGBM-A, MLP-A, RealMLP-A, TabM-A, and TabICLv2-A. For each method the bars show the Elo of the default configuration, the best of 200 random configurations (Tuned), and the greedy ensemble built from the same configurations (Tuned+Ensembled); whiskers are 95% confidence intervals, dashed lines mark the AutoGluon reference systems, and hatched bars are partially imputed methods.

B Setup

B.1 Hardware

Most of the experiments were conducted on a single NVIDIA A100 GPU paired with AMD EPYC CPU. Absolute wall-clock time depends on hardware, parallelism, and cluster load; the agent

[1] Ensemble of LightGBM, CatBoost, XGBoost, random forests, and MLP.

[2] Adds TabPFNv2, TabICL, Mitra, TabM, RealTabPFN-2/2.5, TabDPT, TabPrep-LightGBM, and EBM to the 1.4 ensemble.

Table 6: Overview of the datasets included in the main evaluation; sizes are total row counts. Full per-dataset statistics are in Appendix B.

Source		#Rows				#Features				Task type	
OpenML	TabReD	Min.	Q50	Mean	Max.	Min.	Q50	Mean	Max.	#Regr.	#Classif.
37	8	768	21K	103K	1.2M	5	21	156	1776	20	25

generation times of Table 3 additionally depend on provider load and network latency, so they should be read as indicative rather than reproducible.

B.2 Datasets

We provide a full list of datasets and their properties in Table 7, which also includes the evaluation metric and feature counts. The evaluation metric of each dataset is the native metric adopted from the prior work that introduced the corresponding benchmark (TabM (Gorishniy et al., 2025), TabArena (Erickson et al., 2025), and TabReD (Rubachev et al., 2025)). Table 6 gives the aggregate summary. We categorize datasets by size as follows: the *large* group consists of the TabReD benchmark datasets together with Microsoft; the remaining datasets are split into *small* and *medium* at the median number of cells (rows $\times$ features), with the split boundary at 3.0×10^5 cells.

B.3 Metrics

Each dataset is evaluated with its native metric (Appendix B.2). To compare methods within a dataset, we use the signed score $s_B(m; \mathcal{D})$: the metric value at budget B for lower-is-better metrics (RMSE, log-loss) and its negation for higher-is-better ones (AUROC, accuracy), so that lower s_B is always better. We report four aggregate metrics; as discussed in Erickson et al. (2025), they offer complementary views on the performance.

- **Mean rank.** Pool every (model family, search space) pair on a dataset, sort by s_B, and let $\mathrm{rank}(m, \mathcal{S}; \mathcal{D})$ be the position of $(m, \mathcal{S})$ in the sorted ordering (rank 1 = best). The mean rank averages this across datasets.
- **Relative improvement.** Effect of switching to method m from a reference method r (the tuned base-space MLP†, unless stated otherwise; Table 2 uses the model's own base space as r):

$$\Delta_B(m; \mathcal{D}) = 100 \, \frac{s_B(r; \mathcal{D}) - s_B(m; \mathcal{D})}{|s_B(r; \mathcal{D})|}, \tag{1}$$

  averaged across datasets. Because Δ_B is defined on the signed native metric, its denominator is the metric magnitude itself (e.g. $|\mathrm{AUROC}|$ rather than $|1 - \mathrm{AUROC}|$, which would inflate small absolute changes on near-saturated datasets).
- **Normalized score.** The only metric computed on the error scale: $\mathrm{err} = 1 - \mathrm{Score}$ for the two higher-is-better metrics in our suite, i.e. $\mathrm{err} = 1 - \mathrm{AUROC}$ and $\mathrm{err} = 1 - \mathrm{accuracy}$, while the lower-is-better RMSE and log-loss are used as is, $\mathrm{err} = \mathrm{Score}$. Per dataset, errors are linearly rescaled so that the best method maps to 1, the median method maps to 0, and anything worse than the median is truncated to 0; higher is better (Salinas & Erickson, 2024).
- **Elo rating.** Each per-dataset comparison between two methods is a logistic "match" (the method with the better signed score wins); we fit ratings by minimizing the Bradley–Terry log-loss with L-BFGS over all pairs and dataset/budget combinations following Chiang et al. (2024), then shift the resulting scale so that the MLP† baseline anchors at 1000. Larger Elo means stronger overall performance.

Together, rank and Elo capture the relative standing, while relative improvement and normalized score summarize the scale of the change.

Table 7: Per-dataset statistics. Size is the small/medium/large bucket used in the paper comparisons (large = TabReD datasets and Microsoft; the rest split at the median cell count). Metric is the dataset-native evaluation metric adopted from the source benchmark. #Samples is the total number of rows. For regression we report #Classes as 0.

Dataset	Task	Size	Metric	#Num.	#Cat.	#Bin.	#Classes	#Samples	Source
airfoil_self_noise	reg	small	rmse	4	1	0	0	1,503	OpenML
Another-Dataset-on-used-Fiat-500	reg	small	rmse	6	1	0	0	1,538	OpenML
black-friday	reg	medium	rmse	5	4	0	0	166,821	OpenML
california	reg	small	rmse	8	0	0	0	20,640	OpenML
concrete_compressive_strength	reg	small	rmse	8	0	0	0	1,030	OpenML
diamond	reg	medium	rmse	6	3	0	0	53,940	OpenML
Food_Delivery_Time	reg	medium	rmse	6	3	0	0	45,451	OpenML
healthcare_insurance_expenses	reg	small	rmse	3	3	0	0	1,338	OpenML
house	reg	medium	rmse	16	0	0	0	22,784	OpenML
miami_housing	reg	small	rmse	14	1	0	0	13,776	OpenML
microsoft	reg	large	rmse	136	0	0	0	1,200,192	OpenML
physiochemical_protein	reg	medium	rmse	9	0	0	0	45,730	OpenML
QSAR-TID-11	reg	medium	rmse	1024	0	0	0	5,742	OpenML
QSAR_fish_toxicity	reg	small	rmse	6	0	0	0	907	OpenML
wine_quality	reg	small	rmse	11	1	0	0	6,497	OpenML
cooking-time	reg	large	rmse	186	3	3	0	319,986	TabReD
delivery-eta	reg	large	rmse	221	1	1	0	416,451	TabReD
maps-routing	reg	large	rmse	984	2	0	0	340,981	TabReD
sberbank-housing	reg	large	rmse	365	10	17	0	28,321	TabReD
weather	reg	large	rmse	100	0	3	0	423,795	TabReD
Amazon_employee_access	clf	small	roc_auc	0	9	0	2	32,769	OpenML
APSFailure	clf	medium	roc_auc	170	0	0	2	76,000	OpenML
bank-marketing	clf	medium	roc_auc	5	8	0	2	45,211	OpenML
Bioresponse	clf	medium	roc_auc	1776	0	0	2	3,751	OpenML
churn	clf	small	accuracy	10	1	0	2	10,000	OpenML
credit_card_clients_default	clf	medium	roc_auc	20	3	0	2	30,000	OpenML
customer_satisfaction_in_airline	clf	medium	roc_auc	5	16	0	2	129,880	OpenML
diabetes	clf	small	roc_auc	8	0	0	2	768	OpenML
Diabetes130US	clf	medium	roc_auc	8	39	0	2	71,518	OpenML
E-CommereShippingData	clf	small	roc_auc	6	4	0	2	10,999	OpenML
GiveMeSomeCredit	clf	medium	roc_auc	10	0	0	2	150,000	OpenML
heloc	clf	small	roc_auc	23	0	0	2	10,459	OpenML
HR_Analytics_Job_Change_of_Data_Scientists	clf	small	roc_auc	2	10	0	2	19,158	OpenML
in_vehicle_coupon_recommendation	clf	medium	roc_auc	2	22	0	2	12,684	OpenML
jm1	clf	small	roc_auc	21	0	0	2	10,885	OpenML
kddcup09_appetency	clf	medium	roc_auc	174	38	0	2	50,000	OpenML
NATICUSdroid	clf	medium	roc_auc	0	86	0	2	7,491	OpenML
online_shoppers_intention	clf	small	roc_auc	10	6	1	2	12,330	OpenML
polish_companies_bankruptcy	clf	medium	roc_auc	64	0	0	2	5,910	OpenML
qsar-biodeg	clf	small	roc_auc	36	5	0	2	1,054	OpenML
splice	clf	small	log_loss	0	60	0	3	3,190	OpenML
taiwanese_bankruptcy_prediction	clf	medium	roc_auc	94	0	0	2	6,819	OpenML
ecom-offers	clf	large	roc_auc	113	0	6	2	160,057	TabReD
homecredit-default	clf	large	roc_auc	612	82	2	2	381,664	TabReD
homesite-insurance	clf	large	roc_auc	253	23	23	2	260,753	TabReD

B.4 HYPERPARAMETERS

We use model-specific HPO spaces that match strong published tabular baselines, then compare each baseline against its agent-augmented counterpart under the same split, metric, sampler, and trial budget.

MLP†. Our MLP baseline follows the feed-forward baseline used with TabM: a stack of fully connected blocks with tuned depth, width, dropout, learning rate, weight decay, and piecewise-linear numerical embeddings (Gorishniy et al., 2025; 2022). The PLR embedding bins and embedding dimension are part of the HPO space because numerical embeddings are a major driver of neural tabular performance. The full tuning space is shown in Table 8.

TabM†. TabM is a parameter-efficient neural ensemble that shares most parameters across ensemble members while producing k predictions per example (Gorishniy et al., 2025). We use the mini TabM configuration with $k = 32$, tune the backbone width, depth, dropout, PLR embedding parameters,

Table 8: The hyperparameter tuning space for MLP†.

Parameter	Distribution
# blocks	UniformInt[1, 6]
Block width	UniformInt[64, 1024], step=16
Dropout rate	{0.0, Uniform[0.0, 0.5]}
Learning rate	LogUniform[3e-5, 1e-3]
Weight decay	{0, LogUniform[1e-4, 1e-1]}
Gradient clipping	1.0
Max epochs	256
Early-stopping patience	16
# PLR bins on OpenML	UniformInt[16, 128], step=4
# PLR bins on TabRed, Microsoft	UniformInt[8, 32]
PLR embedding dim	UniformInt[8, 128], step=4
# Optuna iterations / ensemble size on small-medium	200/100
# Optuna iterations / ensemble size on large	100/20

learning rate, and weight decay, and keep the same early-stopping and gradient-clipping protocol as for MLP. The full tuning space is shown in Table 9.

Table 9: The hyperparameter tuning space for TabM†.

Parameter	Distribution
# blocks	UniformInt[2, 4]
Block width	UniformInt[64, 1024]
Dropout rate	Uniform[0.0, 0.5]
Ensemble size k	32
# PLR bins on OpenML	UniformInt[2, 128]
# PLR bins on TabRed, Microsoft	UniformInt[8, 32]
PLR embedding dim on OpenML	UniformInt[8, 32], step=4
PLR embedding dim on TabRed, Microsoft	UniformInt[8, 24], step=4
Learning rate	LogUniform[3e-5, 1e-3]
Weight decay	{0, LogUniform[1e-4, 1e-1]}
Gradient clipping	1.0
Max epochs	256
Early-stopping patience	16
# Optuna iterations / ensemble size on small-medium	200/100
# Optuna iterations / ensemble size on large	100/20

RealMLP. RealMLP is included as a strong pre-tuned modular MLP baseline designed to be competitive without extensive manual retuning (Holzmüller et al., 2024). We tune the RealMLP choices exposed in its recommended configuration family, including front scaling, dropout, activation, hidden-size template, numerical embedding type, PLR scale, learning rate, weight decay, and label smoothing for classification. The full tuning space is shown in Table 10.

LightGBM. LightGBM is a histogram-based gradient-boosted decision tree method (Ke et al., 2017). We use the HPO ranges adopted by the TabR benchmark protocol (Gorishniy et al., 2024), tuning feature and bagging fractions, learning rate, leaf count, L2 regularization, and the minimum Hessian mass per leaf. The number of estimators is fixed at 4000 with early stopping after 200 rounds. The full tuning space is shown in Table 11.

Table 10: The hyperparameter tuning space for RealMLP.

Parameter	Distribution / value
Large configuration flag	Categorical{False, True}
Hidden layout	`rectangular`
# hidden layers	IntUniform[2, 4]
Hidden width	Categorical{256, 384, 512}
Activation	`mish`
Dropout p	Uniform[0.0, 0.5]
Dropout schedule	`flat_cos`
Embedding size	Categorical{4, 8, 16}
PLR σ	LogUniform[10^{-2}, 50]
PLR learning-rate factor	LogUniform[$5 \cdot 10^{-2}$, $3 \cdot 10^{-1}$]
PLR hidden widths (for large)	Categorical{8, 16, 32, 64}x2
PLR hidden widths (for not large)	(16, 4)
Max one-hot category size	$\lfloor \text{LogUniform}[4, 33] \rfloor$
Learning rate	LogUniform[$2 \cdot 10^{-2}$, $3 \cdot 10^{-1}$]
Weight decay	LogUniform[10^{-3}, $5 \cdot 10^{-2}$]
Scale learning-rate factor	LogUniform[2, 10]
First-layer learning-rate factor	LogUniform[0.3, 1.5]
$1-$ squared momentum	LogUniform[$5 \cdot 10^{-3}$, $5 \cdot 10^{-2}$]
Use label smoothing	Categorical{False, True}
Label smoothing ϵ	LogUniform[$5 \cdot 10^{-3}$, 10^{-1}]
Label smoothing schedule	`coslog4`
Epochs (for large)	Categorical{256, 512}
Epochs (for not large)	256
Early stopping (for large)	(3, 40) = (mult., add.)
Early stopping (for not large)	disabled
Ensemble size	8
Ensemble averaging before softmax	False
# Optuna iterations / ensemble size on small-medium	200/100
# Optuna iterations / ensemble size on large	100/20

Table 11: The hyperparameter tuning space for LightGBM.

Parameter	Distribution
feature_fraction	Uniform[0.5, 1.0]
learning_rate	LogUniform[1e-3, 1.0]
num_leaves	UniformInt[4, 768]
min_sum_hessian_in_leaf	LogUniform[1e-4, 100]
bagging_fraction	Uniform[0.5, 1.0]
lambda_l2	{0, LogUniform[0.1, 10.0]}
n_estimators	4000 (fixed)
bagging_freq	1
early_stopping_rounds	200
# Optuna iterations / ensemble size on small-medium	200/100
# Optuna iterations / ensemble size on large	100/20

TabICLv2. TabICLv2 is a tabular foundation model for in-context learning on large tabular datasets (Qu et al., 2026). We keep the published `TabICLv2` checkpoint fixed and do not train or fine-tune the backbone. We fit the model by passing the concatenated train and validation portions of the dataset as the context: since TabICLv2 performs no gradient-based training, the validation split is

not needed for early stopping and would otherwise go unused, whereas the other families consume it for early stopping and model selection. To avoid out-of-memory issues, we subsample the context to the minimum of $40,000$ samples or $3,840,000/n_{\text{features}}$.

B.5 Hyperparameter optimization and ensembling

Hyperparameter optimization. We use the univariate TPE sampler in Optuna to tune individual model hyperparameters (Bergstra et al., 2011; Akiba et al., 2019). For small and medium datasets, we set the budget to 200 trials; for large ones (TabReD and Microsoft), to 100 trials. The exception is TabICLv2: the default model is evaluated as published, without any tuning, and the agentic TabICLv2 is tuned with a fixed budget of 100 trials on all datasets. In both cases, the first 20 trials use the random sampler to encourage exploration of the space. Empirically, we find the default univariate TPE sampler with the above budgets to work well, and we leave the search for better defaults to future work. For example, the number of random startup trials could be chosen adaptively based on the Coupon Collector's problem (Tang).

Ensembles. We build greedy ensembles following Caruana et al. (2004). We first sample 100 random configurations (20 for large datasets) from the given HPO space and train a model for each. We then iteratively grow the ensemble for a fixed budget of steps: at each step, we try adding each fitted model (selection is with replacement), average its predictions with those of the already selected models, and keep the candidate that yields the lowest validation error. Note that we do not use early stopping: the ensemble is grown for the full budget even if the validation error stops improving at some intermediate step.

C Future Research Directions

Beyond the aggregate gains, the per-axis selection patterns in Figure 3 reveal a structural pattern in how tabular research progresses. On three of the four most important modules (the model, the training procedure, and the numerical preprocessing), the non-default agent-proposed implementations are selected most often. The fourth module, numerical embeddings, is the only one whose default remains competitive, and it is also the one that has received by far the most attention in the recent tabular deep learning (Gorishniy et al., 2022; 2025; Holzmüller et al., 2024). The other three modules, by comparison, remain underexplored: while isolated efforts exist – for instance, recent work on optimizers for tabular deep learning (Gorishniy et al., 2026) – much else is overlooked. Therefore, we believe more investigation into the other pipeline modules is needed.

We attribute this pattern to two complementary factors. First, there is a long tail of plausible-but-forgotten methods on each underexplored axis, displaced, we suspect, by inertia rather than by evidence. Second, the breadth of the hypothesis pool available to an agent far exceeds that of a human researcher: an agent can inexpensively "swarm" the literature for plausible candidates whenever the user requests it, with no commitment to championing any single one. On the other hand, researchers and machine learning practitioners tend to concentrate on a few well-established options. The combination – a long tail of forgotten methods and a wide retrieval window – is, in our view, the principal reason agent-augmented HPO improves over fixed search spaces precisely on the underexplored axes. We view a systematic study of such overlooked axes – now cheap to enumerate via agentic generation – as a productive direction in itself.

D Agent Setup and Artifacts

Between the runs and before the first use, we sanitize the agent environment and memory to avoid potential leaks or biases by running the following procedure:

- Remove memory and session information via
  - `rm -rf /.claude/projects/<...>/memory`
  - `rm -rf /.claude/projects/<...>/sessions`
- Checkout a clean branch: `git checkout -b <semiagent-version-x>`
- Sanitize `git` history

Table 12: Interchangeable components of the two neural AutoML libraries and of the agentic MLP† space, mapped onto our module axes; counts exclude the default implementation on each axis. Axes dispatched by task are reported as regression/classification.

Module axis	AutoPyTorch 0.2.1	AutoKeras 1.1.0	Agentic MLP†
Numerical preprocessing	22	1	7
Categorical preprocessing	2	0	3
Target preprocessing	0	0	4/0
Numerical embedding	0	0	11
Categorical embedding	1	0	8
Architecture	8	1	12
Training	7	0	11
Optimizer	4	2	1
Loss	0	0	2/1
Inference	0	0	2/3
Total	44	4	61/57

- `git reflog expire –expire=now –all`
- `git gc –prune=now –quiet`

D.1 AutoML baselines

Protocol. We run AutoKeras 1.1.0 (TensorFlow 2.15), AutoPyTorch 0.2.1, and AutoGluon 1.5.0 unmodified on the 45 datasets with our exact test splits. We pass our validation split directly into the system: AutoPyTorch via a custom holdout, AutoKeras via `validation_data`). AutoGluon performs its own internal bagging, so we pass it concatenated train and validation subsets. We set the budget to 4 hours for each system following Erickson et al. (2025); Gijsbers et al. (2024). AutoPyTorch returns both the best single model and the greedy ensemble. AutoKeras returns the best model. AutoGluon returns a single predictor, which is a bagged and stacked ensemble.

Library size. Table 12 maps the search spaces of the two neural AutoML systems onto our module axes and counts interchangeable implementations, excluding the default on each axis. AutoPyTorch's 22 numerical-preprocessing components are 7 scalers and the variance threshold plus 15 feature-preprocessing transforms it inherits from auto-sklearn (Feurer et al., 2015); its 8 architecture components are 4 backbones and 4 weight initializers, and its 7 training components are 6 learning-rate schedulers and MixUp. AutoKeras exposes normalization and batch normalization as on/off switches and two optimizer alternatives; the rest of its structured-data space is numeric (depth, width, dropout, learning rate).

AutoGluon portfolio. Under `best_quality`, AutoGluon fits a fixed, meta-learned zero-shot portfolio of 110 configurations (Table 13), bags each configuration with 8-fold cross-validation, feeds the out-of-fold predictions to a second stack layer of the same model families, and combines the resulting models with greedy weighted ensembling. In the `extreme_quality` preset, AutoGluon replaces the portfolio of 110 with a portfolio of 28 configurations which also includes TabM and tabular foundation models (RealTabPFN-v2, TabDPT, TabICL, Mitra).

Agentic hand-off (AutoGluon (+A)). We add 8 randomly-sampled configurations of the agentic RealMLP and TabM† models. The additional agentic configurations are interleaved with the stock portfolio and are bagged, stacked, and ensembled exactly like the stock members under the same four-hour budget (126 configurations in total).

D.2 Human-curated module bank

Motivation. The AutoML comparison of Section 4.1 confounds the module library with the system around it: AutoKeras and AutoPyTorch differ from our pipelines in base model, tuner, and ensembler. To isolate the library, we build a human-curated bank from a published search space and run it through our own pipelines. We take AutoPyTorch 0.2.1 (Zimmer et al., 2021), the only system in Table 12 that modularizes the network itself, and port every component into one of our module axes (see

Table 13: Composition of AutoGluon's zero-shot portfolios: `best_quality` (v1.5.0, as resolved in our runs) and `extreme_quality` (v1.5, portfolio `zeroshot_2025_12_18_gpu`). Entries are the number of configurations per model family.

AutoGluon key	Model family	`best_quality`	`extreme_quality`
`GBM`	LightGBM	16	5
`GBM_PREP`	TabPrep-LightGBM	–	5
`CAT`	CatBoost	20	1
`XGB`	XGBoost	10	–
`RF`	Random forest	10	–
`XT`	Extra trees	10	–
`NN_TORCH`	PyTorch MLP	21	–
`FASTAI`	FastAI tabular NN	23	–
`TABM`	TabM	–	5
`REALTABPFN-V2`	RealTabPFN-v2	–	5
`TABDPT`	TabDPT	–	5
`TABICL`	TabICL	–	1
`MITRA`	Mitra	–	1
Total		110	28

Table 14: Human-curated module bank vs. agent-generated bank on the curated 45 datasets. The human bank ports the 44 components of AutoPyTorch into the same MLP†, TabM†, and RealMLP pipelines as the agentic bank, and both are tuned and ensembled with the identical protocol (Section 3.1). Δ_{base}: mean relative improvement (%) over the family's base space on the dataset-native metric, with a bootstrap 95% CI over datasets. Wins: datasets on which the space beats its base space; last column: datasets on which the agentic space beats the human bank.

		Human bank (AutoPyTorch)		Agentic		Agentic vs. bank
Family	Scope	Δ_{base} (%) ↑	Wins	Δ_{base} (%) ↑	Wins	Wins
MLP†	Single	$-1.6\,[-5.8, +0.7]$	20/45	$+1.7\,[+0.5, +3.2]$	35/45	37/45
	Ensemble	$+0.2\,[-0.6, +1.1]$	19/45	$+1.8\,[+0.9, +3.1]$	38/45	39/45
TabM†	Single	$-0.6\,[-1.9, +0.5]$	15/45	$+0.6\,[-0.4, +1.7]$	35/45	38/45
	Ensemble	$+0.6\,[-0.6, +1.8]$	21/45	$+1.5\,[+0.5, +2.7]$	37/45	38/45
RealMLP	Single	$+0.6\,[-0.2, +1.6]$	20/45	$-0.0\,[-0.6, +0.5]$	21/45	25/45
	Ensemble	$-0.5\,[-1.3, +0.0]$	15/45	$+0.3\,[-0.2, +0.9]$	30/45	32/45

module counts in Table 12). Each component is wrapped exactly like an agent-generated module: one categorical index per axis, the shared base hyperparameters are read, and every component-internal setting is pinned to the package default, so no conditional hyperparameters enter the space.

Protocol. The bank is plugged into the MLP†, TabM†, and RealMLP pipelines and tuned with the protocol of Section 3.1: TPE with 200 trials (100 on large datasets), 15 evaluation seeds for the best configuration, and a greedy ensemble over 100 random configurations (20 on large datasets). We report the results on our curated-bank of 45 datasets.

Results. Table 14 compares both banks against the base space of each family on the same datasets. For MLP† and TabM† the human bank does not improve the tuned single model (-1.6% and -0.6%); head to head, the agentic space beats the bank on 37 / 39 of the 45 datasets for MLP† and on 38 / 38 for TabM† in tuned / ensembled regimes. For RealMLP, whose base recipe already fixes a strong training procedure, neither bank changes the tuned single model beyond noise, and only the agentic bank improves the ensemble ($+0.3\%$ vs. -0.5%). Extending the search space with a curated library of standard components therefore does not by itself reproduce the gain. Part of the difference is the size and diversity: the human-curated bank is smaller and contains fewer recent works (e.g. numerical embeddings (Gorishniy et al., 2022), Lion (Chen et al., 2023), SAM (Foret et al., 2021)).

```
bin/modules/
+-- embedding/
|   +-- __init__.py
|   +-- one_hot.py
+-- loss_reg/
    +-- __init__.py
    +-- huber.py
```

Figure 6: Module registry package structure under `bin/modules`. Each subdirectory corresponds to one pipeline stage; individual `.py` files implement registered module choices for that axis.

D.3 HYPOTHESIS-GENERATION WORKFLOW

Agent environment. Each model family is handled by a single agent run. The agent is given the repository for that family: a small pipeline in which each module axis (preprocessing, embedding, model, training, optimizer, loss, and inference) is represented by a package whose registry maps integer indices to builder functions (Figure 6). The agent is equipped with two main tools: `inspect_pipeline.py`, which prints the registered modules across all axes, and `test.py`, which runs a synthetic-data smoke test for a chosen module combination.

Workflow. After a preflight check that the environment is set up and the baseline runs, the agent proceeds in four stages: *inspection*, *hypothesis generation*, *implementation*, and *testing*. During *inspection*, the agent maps out the repository structure and the model pipeline. During *hypothesis generation*, it proposes up to `--num-hypotheses` candidate ideas, spread across the axes, and filters out those that do not meet our evidence bar: an idea must (a) have a clear mechanism by which it affects the model, (b) have support in the literature, (c) not replicate something already present in the base model, and (d) be benign by default, i.e., not harm performance when inactive. The surviving hypotheses are then implemented one at a time in an implement-and-test loop. For each module, the agent: (1) implements it as a new self-contained `.py` file; (2) registers it in the package's `__init__.py`; (3) runs the smoke test (`uv run test.py --<axis>-idx <idx>`); (4) reads any performance and efficiency warnings; and (5) debugs the module, or drops it if it degrades runtime or memory by more than 10x times. After all hypotheses are processed, the agent tests module *interactions* by running `uv run test.py --exhaustive` and debugs or discards any problematic combinations. The full candidate sets are provided in Tables 16–20.

Synthetic dataset test. The `test.py` script evaluates each candidate against the all-`v0` baseline on 4 temporary synthetic datasets generated inside the test runner. Each scenario has $N = N_{\text{train}} + N_{\text{val}} + N_{\text{test}}$ rows, p numerical features, and q categorical features, with $(p, q) \in \{(100, 100), (12, 12), (16, 0), (0, 16)\}$. For numerical features, the runner samples

$$x_{ij}^{\text{num}} = z_{ij} + t_i a_j, \qquad z_{ij} \sim \mathcal{N}(0, 1),$$

where t_i is linearly spaced from -1 to 1 over rows and a_j is linearly spaced from 0.05 to 0.5 over numerical columns. Categorical features are sampled as integer codes. By default,

$$x_{ij}^{\text{cat}} \sim \text{Unif}\{0, \ldots, 5\}.$$

In the $(12, 12)$ categorical edge-case scenario, the first four categorical columns use

$$x_{ij}^{\text{cat}} \sim \text{Unif}\{0, \ldots, 3\} \quad \text{on train}, \qquad x_{ij}^{\text{cat}} \sim \text{Unif}\{0, \ldots, 4\} \quad \text{on validation/test},$$

with category 4 forced to appear outside training. The next four columns use the converse stress case,

$$x_{ij}^{\text{cat}} \sim \text{Unif}\{0, \ldots, 6\} \quad \text{on train}, \qquad x_{ij}^{\text{cat}} \sim \text{Unif}\{0, \ldots, 2\} \quad \text{on validation/test},$$

with category 6 forced to appear in training. Remaining categorical columns use $\text{Unif}\{0, \ldots, 4\}$ in all splits.

Targets are generated from a finite signal plus noise. Let

$$s_i = \sum_{j=1}^{\min(p,8)} b_j x_{ij}^{\text{num}} + \sum_{j=1}^{\min(q,8)} c_j (x_{ij}^{\text{cat}} \bmod 7) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}(0, 0.1^2),$$

where b_j is linearly spaced from 0.2 to 1.0 and c_j from 0.03 to 0.11. Regression targets are $y_i = s_i$. Binary labels are $y_i = \mathbf{1}\{s_i > \text{median}_{k \in \text{train}}(s_k)\}$. Multiclass labels are obtained by discretizing s_i at the training-set tertiles. The runner uses a short two-epoch pass by default and a twelve-epoch long pass to expose modules that activate only after warmup or several epochs, such as schedulers, exponential moving averages (Polyak & Juditsky, 1992), or training-loop changes. It checks that predictions and metrics are finite, and reports wall time, iteration speed, trainable parameter count, process memory, and GPU memory when CUDA is used.

D.4 AUTORESEARCH DETAILS

We adapt the autoresearch repository framework for tabular data and describe its mechanics below (Karpathy, 2026). Figure 7 shows the per-iteration trajectories on the four evaluated datasets. In Table 15, we compare the autoresearch best proposed solution to the matched trial-budget solutions obtained by the agentic and classical HPO. We see that the classical HPO matches or beats autoresearch on three of the four datasets, while agentic HPO further improves on the classical version everywhere. We note that due to the cost (varying from 20 to 50 USD per run), we do not re-run the autoresearch, so we report a single run without error bars.

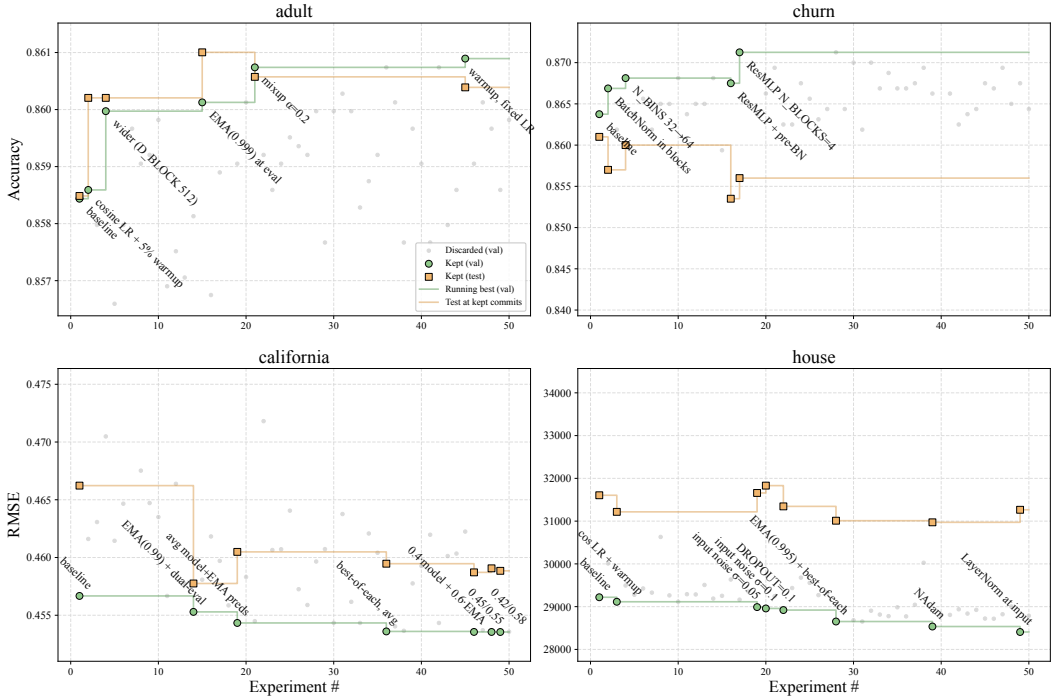


Figure 7: Autoresearch on MLP†: 50 iterations on four datasets. Validation improves only marginally and trails classical HPO and our agentic HPO; test does not consistently follow validation.

Table 15: Comparison of Autoresearch, Default HPO and Agentic HPO at a budget of 50 trials each on the MLP† pipeline.

Method	adult ↑	churn ↑	california ↓	house ↓
Autoresearch	0.8604	0.8560	0.4589	31,265
Default HPO	0.8626 ± 0.0011	0.8599 ± 0.0018	0.4584 ± 0.0036	$31{,}361 \pm 356$
Agentic HPO	0.8671 ± 0.0011	0.8621 ± 0.0031	0.4532 ± 0.0030	$30{,}335 \pm 201$

Unlike our method, which uses the agent only to propose candidate module implementations, the autoresearch baseline of Section 3.2 lets the agent edit the model end to end. It runs an autonomous loop on a single regression or classification task. Starting from the baseline `train.py`, each step (i)

edits `train.py`—any hyperparameter, preprocessing, embedding, optimizer, loss, training-loop, or architecture change is allowed; (ii) commits the change; (iii) runs `benchmark.py`, which prints a single validation score; and (iv) keeps the commit if validation improves and reverts it (`git reset --hard`) otherwise, appending every trial to a results log. The data loader `prepare.py` and the entry point `benchmark.py` are read-only, the held-out test split is never exposed, and the model must remain a single network of the corresponding family (no model swap, no ensembling of independently trained models, no new dependencies). The loop runs for a fixed iteration count. Because it edits one dataset at a time and accepts changes greedily on validation, it neither amortizes generation across datasets nor captures the module interactions that joint HPO exploits, which is reflected in its weaker results in Table 15.

Table 16: Agent-proposed module candidates for MLP† (67 candidates, deduplicated across the regression and classification branches). Counts exclude the default `v0` module on each axis.

Axis	Candidate variants
Numerical preprocessing	Engineered features, power/Yeo–Johnson, RankGauss, row statistics, standardization, uniform-quantile, winsorization, trimmed-MAD (Yeo & Johnson, 2000)
Categorical preprocessing	Hashing, rare-bucketing, target-ordered encoding (Weinberger et al., 2009; Micci-Barreca, 2001)
Target preprocessing	Log, MAD, quantile, symlog target transforms (regression)
Numerical embedding	Linear, piecewise-linear, PLR (learnable, normed, tree, rank, per-feature MLP, Fourier-mapped), B-spline, periodic, soft-binning, bilinear (Gorishniy et al., 2022; Rahimi & Recht, 2007)
Categorical embedding	One-hot, one-hot+frequency, learned/entity, sqrt-sized entity, frequency-weighted, binary-coded, hashed, target-mean (Guo & Berkhahn, 2016; Weinberger et al., 2009)
Architecture	Bottleneck, feature-dropout, GELU, GLU, pre-norm, residual, scaled-init, squeeze–excitation, spectral-norm, wide, wide–narrow, weight-standardized GELU (Dauphin et al., 2017; Shazeer, 2020; He et al., 2016; Hu et al., 2018)
Training	Cosine and warm-restart schedules, EMA, SWA, model soup, snapshot ensemble, gradient noise, MixUp, swap-noise, stochastic weight perturbation (Loshchilov & Hutter, 2017; Polyak & Juditsky, 1992; Izmailov et al., 2018; Zhang et al., 2018)
Optimizer	Lookahead wrapper (Zhang et al., 2019)
Loss	Huber and Gaussian-NLL regression losses; label smoothing (classification) (Huber, 1964; Szegedy et al., 2016)
Inference	Train-range clipping, MC-dropout (regression); Gaussian-noise TTA, temperature scaling (classification) (Gal & Ghahramani, 2016; Guo et al., 2017)

Table 17: Agent-proposed module candidates for TabM† (Gorishniy et al., 2025) (56 candidates). Counts exclude the default `v0` module on each axis.

Axis	Candidate variants
Numerical preprocessing	Clipped-quantile, missing indicators, log1p, robust/IQR, standardization, k-means-augmented quantile, rank-uniform, Yeo–Johnson (Yeo & Johnson, 2000; MacQueen, 1967)
Categorical preprocessing	Frequency encoding, hashing (Weinberger et al., 2009)
Target preprocessing	Log1p, quantile, robust target transforms (regression)
Numerical embedding	Advanced PLR, gated PLR, piecewise-linear, polynomial, tokenized (Gorishniy et al., 2022)
Categorical embedding	One-hot, one-hot+learned, one-hot with unknown bucket, learned, target-mean (Micci-Barreca, 2001)
Architecture	Bilinear, column/feature dropout, FiLM, GeGLU, SwiGLU+RMSNorm, input gates, linear-residual, LoRA, pre-norm, squeeze–excitation, soft-MoE, sparse-hidden, wide-shallow, categorical-embedded (Perez et al., 2018; Shazeer, 2020; Zhang & Sennrich, 2019; Hu et al., 2022; Puigcerver et al., 2024)
Training	Cosine schedule, CutMix, EMA, MixUp, negative-correlation, SAM, SWA, head decorrelation (Polyak & Juditsky, 1992; Zhang et al., 2018; Yun et al., 2019; Foret et al., 2021; Izmailov et al., 2018)
Optimizer	Lion (Chen et al., 2023)
Loss	Huber regression loss (Huber, 1964)
Inference	Clipped, geometric, median, trimmed-mean, quantile-average, variance-weighted head aggregation, Gaussian-noise TTA (regression); temperature scaling (classification) (Lakshminarayanan et al., 2017; Guo et al., 2017)

Table 18: Agent-proposed module candidates for RealMLP (Holzmüller et al., 2024) (13 candidates). Counts exclude the default `v0` module on each axis.

Axis	Candidate variants
Numerical preprocessing	Normal-quantile, Yeo–Johnson power transform (Gorishniy et al., 2021; Yeo & Johnson, 2000)
Categorical preprocessing	Rare-category grouping, log-frequency count encoding
Numerical embedding	Piecewise-linear embedding (Gorishniy et al., 2022)
Categorical embedding	Shared-vocabulary embeddings (Gorishniy et al., 2025)
Architecture	BatchEnsemble layers, pre-activation residual network (Wen et al., 2020; He et al., 2016)
Training	Exponential moving-average (EMA) weights, MixUp augmentation (Polyak & Juditsky, 1992; Zhang et al., 2018)
Optimizer	Global-norm gradient clipping (Pascanu et al., 2013)
Loss	Focal loss (classification); Huber loss (regression) (Lin et al., 2017; Huber, 1964)

Table 19: Agent-proposed module candidates for LightGBM (Ke et al., 2017) (48 candidates). The base learner is fixed; candidates act before (feature, augmentation) or after (post-processing) the learner. Counts exclude the default `v0` module on each axis.

Axis	Candidate variants
Feature engineering	Target (k-fold) / frequency / groupby-mean encoding, categorical pair-concat and target-std, high-cardinality hashing, k-means cluster/distance, PCA, random projection, pairwise products, pairwise differences, quantile-bin, rank-percentile, row statistics, log1p-skew, NaN indicators/row-count, rare-category grouping, extreme counts, top-k squares (Micci-Barreca, 2001; MacQueen, 1967)
Data augmentation	(Stratified) bootstrap, Gaussian noise, uniform/rank jitter, column-swap and intra-target-swap noise, categorical/feature dropout, mean-imputation dropout, MixUp, minority oversampling, class undersampling, random duplication, low-variance swap, target-quantile bootstrap, perturbed-copy concat, outlier trimming, within-class interpolation (Efron, 1979; Zhang et al., 2018)
Post-processing	Train-range/quantile clipping and renormalization, IQR winsorization, soft-clip, integer rounding, non-negativity clip, probability floor/clip, power sharpen/smooth, temperature scaling (smoothed), prior correction, prior-blend, train-prior balancing, quantile matching, std-matching, mean shrinkage, snap-to-uniques, label smoothing (Guo et al., 2017; Szegedy et al., 2016)

Table 20: Agent-proposed module candidates and implementation baskets for TabICLv2 (Qu et al., 2025; 2026) (standard module pool: 14). The foundation-model backbone is frozen, so candidates act on inputs (preprocessing, permutation, augmentation) or outputs (ensembling, post-processing). Basket rows indicate module baskets used for ensembles.

Axis	Candidate variants / basket choices
Feature preprocessing	Power, quantile-normal, quantile-uniform, quantile (no standardization) (Gorishniy et al., 2021)
Feature permutation	Latin-square, random, and shift feature permutations (Qu et al., 2025)
Data augmentation	Feature-noise (Qu et al., 2026)
Ensembling	Mean, median, geometric-mean, and uncertainty-weighted (Caruana et al., 2004; Lakshminarayanan et al., 2017)
Post-processing	log1p, standardize (regression) (Guo et al., 2017)
Feature-preproc baskets	**0**: identity + power; **1**: identity + power + quantile-normal + quantile-uniform; **2**: identity + power + raw quantile-normal
Augmentation baskets	**0**: identity / no augmentation; **1**: identity + feature-noise augmentation
Postproc baskets (clf.)	**0**: probability readout; **1**: probability readout + temperature scaling; **2**: probability readout + prior-blend
Postproc baskets (reg.)	**0**: standardize; **1**: standardize + log1p

E Extended Results

This appendix reports extended results. The default variant searches only the model-specific space $\mathcal{S}_{\mathrm{HPO}}$ described in Appendix B; the agentic variant searches the augmented space $\mathcal{S}_{\mathrm{agent}}$, which contains the default module choices as a subset. We provide the full list of agent-generated hypotheses in Tables 16–20.

For each model family, the agent-generated candidate set is produced once and then reused across all datasets. The agent does not receive dataset contents or dataset-specific feedback during generation. Downstream selection is performed only by HPO: each trial is trained on the training split, selected by validation performance, and evaluated on the held-out test split after selection. We report the mean and standard deviation of the test metric over repeated evaluation seeds.

Table 21: Per-run results behind Table 4: each row is one agent-generated MLP† search space evaluated on all 45 datasets in the joint 12-space pool. Win rate is the fraction of datasets on which the run improves over the tuned base-space MLP†.

Run	Mean rank ↓	Norm. score ↑	Elo ↑	Improv. over MLP† (%) ↑	Win rate (%) ↑
Released (Claude)	6.96	0.25	1095	+1.28	60
Claude run 1	6.64	0.21	1114	+0.43	64
Claude run 2	5.78	0.28	1165	+0.33	76
Claude run 5	6.16	0.26	1142	+1.28	69
Claude run 6	5.49	0.35	1182	+1.39	73
Claude run 7	7.60	0.15	1056	+0.49	60
Codex run 0	5.73	0.32	1168	+0.59	73
Codex run 1	6.18	0.29	1141	+0.62	69
Codex run 2	6.56	0.22	1119	+1.22	67
Codex run 3	6.24	0.24	1137	+1.16	69
Codex run 4	6.18	0.22	1141	+1.27	69

E.1 Ablation study

Table 21 reports the per-run agent results behind the summary in Section 5.2. All runs use the agentic MLP† pipeline under the same split, metric, validation rule, and trial budget as the main experiments.

E.2 Full per-dataset results

We report full per-dataset results extracted at the terminal budget: 200 trials for small and medium datasets and 100 trials for the large ones in Table 22. Below the family rows, each subtable also lists the individual-model rows of AutoPyTorch and AutoKeras; these are single unseeded runs, so no standard deviation is reported.

Table 22: Individual results for each of the 45 datasets. Each subtable reports the terminal-budget mean test metric ± standard deviation for Default HPO and Agentic HPO under the same split, metric, and HPO budget. Arrows indicate whether higher or lower values are better.

black-friday ($\times 10^3$) ↓

Model	Default HPO	Agentic HPO
MLP	3.4728 ± 0.0025	3.4779 ± 0.0034
RealMLP	3.4497 ± 0.0019	3.4369 ± 0.0010
LightGBM	3.4516 ± 0.0005	3.4501 ± 0.0004
TabM	3.4551 ± 0.0032	3.4484 ± 0.0018
TabICL	3.5368 ± 0.0034	3.5344 ± 0.0020
AutoPyTorch	3.5426	
AutoKeras	3.6116	

california ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	4.5350 ± 0.0199	4.3717 ± 0.0188
RealMLP	4.0618 ± 0.0222	4.0709 ± 0.0179
LightGBM	4.3319 ± 0.0153	4.3344 ± 0.0172
TabM	4.3025 ± 0.0228	4.1363 ± 0.0212
TabICL	3.9784 ± 0.0062	3.8974 ± 0.0193
AutoPyTorch	5.0470	
AutoKeras	5.0945	

churn ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.6110 ± 0.0275	8.6003 ± 0.0301
RealMLP	8.6223 ± 0.0133	8.6220 ± 0.0171
LightGBM	8.6033 ± 0.0092	8.5923 ± 0.0216
TabM	8.6210 ± 0.0124	8.6143 ± 0.0226
TabICL	8.6407 ± 0.0147	8.6500 ± 0.0140
AutoPyTorch	8.5500	
AutoKeras	8.5200	

diamond ($\times 10^2$) ↓

Model	Default HPO	Agentic HPO
MLP	5.2571 ± 0.0280	5.2047 ± 0.0321
RealMLP	5.2679 ± 0.0199	5.1885 ± 0.0200
LightGBM	5.3933 ± 0.0113	5.3420 ± 0.0206
TabM	5.2084 ± 0.0145	5.1835 ± 0.0121
TabICL	5.0034 ± 0.0125	4.9983 ± 0.0071
AutoPyTorch	5.5970	
AutoKeras	5.9986	

house ($\times 10^4$) $\downarrow$

Model	Default HPO	Agentic HPO
MLP	3.0997 ± 0.0385	3.0567 ± 0.0429
RealMLP	3.0864 ± 0.0286	3.0734 ± 0.0302
LightGBM	3.1362 ± 0.0031	3.1339 ± 0.0057
TabM	3.0441 ± 0.0110	2.9945 ± 0.0101
TabICL	2.7962 ± 0.0064	2.8235 ± 0.0000
AutoPyTorch	3.1727	
AutoKeras	3.3025	

microsoft ($\times 10^{-1}$) $\downarrow$

Model	Default HPO	Agentic HPO
MLP	7.4654 ± 0.0036	7.4610 ± 0.0040
RealMLP	7.4176 ± 0.0030	7.4350 ± 0.0017
LightGBM	7.4159 ± 0.0011	7.4162 ± 0.0011
TabM	7.4143 ± 0.0020	7.4285 ± 0.0020
TabICL	7.6311 ± 0.0248	7.5985 ± 0.0129
AutoPyTorch	7.7543	
AutoKeras	7.6362	

tabarena/airfoil_self_noise $\downarrow$

Model	Default HPO	Agentic HPO
MLP	1.2111 ± 0.0851	1.0652 ± 0.0188
RealMLP	1.1872 ± 0.0386	1.1341 ± 0.0186
LightGBM	1.6071 ± 0.0331	1.3727 ± 0.0110
TabM	1.0746 ± 0.0327	1.1152 ± 0.0335
TabICL	0.9991 ± 0.0103	1.0006 ± 0.0119
AutoPyTorch	1.9506	
AutoKeras	2.1980	

tabarena/Amazon_employee_access ($\times 10^{-1}$) $\uparrow$

Model	Default HPO	Agentic HPO
MLP	8.3297 ± 0.0155	8.3507 ± 0.0356
RealMLP	8.4477 ± 0.0540	8.5627 ± 0.0634
LightGBM	8.2752 ± 0.0388	8.3810 ± 0.0258
TabM	8.3070 ± 0.0772	8.3013 ± 0.0486
TabICL	8.4644 ± 0.0119	8.4433 ± 0.0116
AutoPyTorch	8.3501	
AutoKeras	6.0564	

tabarena/Another-Dataset-on-used-Fiat-500 ($\times 10^2$) $\downarrow$

Model	Default HPO	Agentic HPO
MLP	7.3823 ± 0.0957	7.1248 ± 0.0164
RealMLP	7.4757 ± 0.1191	7.5793 ± 0.1223
LightGBM	7.1827 ± 0.0270	7.1960 ± 0.0745
TabM	7.2767 ± 0.0315	7.1903 ± 0.0398
TabICL	6.8200 ± 0.0197	6.8626 ± 0.0056
AutoPyTorch	7.1934	
AutoKeras	7.3005	

tabarena/APSFailure ($\times 10^{-1}$) $\uparrow$

Model	Default HPO	Agentic HPO
MLP	9.9218 ± 0.0064	9.8721 ± 0.0307
RealMLP	9.9309 ± 0.0096	9.9140 ± 0.0140
LightGBM	9.9287 ± 0.0092	9.9302 ± 0.0080
TabM	9.9330 ± 0.0062	9.9426 ± 0.0031
TabICL	9.9370 ± 0.0017	9.9389 ± 0.0042
AutoPyTorch	9.9031	
AutoKeras	9.8351	

tabarena/bank-marketing ($\times 10^{-1}$) $\uparrow$

Model	Default HPO	Agentic HPO
MLP	7.7260 ± 0.0052	7.6111 ± 0.0182
RealMLP	7.6694 ± 0.0149	7.6746 ± 0.0199
LightGBM	7.7145 ± 0.0166	7.6781 ± 0.0194
TabM	7.7285 ± 0.0056	7.6818 ± 0.0093
TabICL	7.1941 ± 0.3702	7.4738 ± 0.1351
AutoPyTorch	7.4950	
AutoKeras	7.5013	

tabarena/Bioresponse ($\times 10^{-1}$) $\uparrow$

Model	Default HPO	Agentic HPO
MLP	8.5302 ± 0.0492	8.5757 ± 0.0368
RealMLP	8.6958 ± 0.0415	8.6243 ± 0.0370
LightGBM	8.7933 ± 0.0377	8.6900 ± 0.0279
TabM	8.7259 ± 0.0186	8.6201 ± 0.0194
TabICL	8.6475 ± 0.0267	8.6487 ± 0.0116
AutoPyTorch	8.2180	
AutoKeras	8.2771	

tabarena/concrete_compressive_strength $\downarrow$

Model	Default HPO	Agentic HPO
MLP	4.8574 ± 0.1416	4.6435 ± 0.1284
RealMLP	4.6979 ± 0.0916	4.6595 ± 0.0599
LightGBM	4.7099 ± 0.0469	4.6605 ± 0.1111
TabM	5.0098 ± 0.1014	4.3319 ± 0.0715
TabICL	3.6744 ± 0.0270	3.6683 ± 0.0307
AutoPyTorch	5.3965	
AutoKeras	4.9451	

tabarena/credit_card_clients_default ($\times 10^{-1}$) $\uparrow$

Model	Default HPO	Agentic HPO
MLP	7.8489 ± 0.0278	7.8524 ± 0.0232
RealMLP	7.9047 ± 0.0109	7.8049 ± 0.0526
LightGBM	7.8778 ± 0.0116	7.8909 ± 0.0187
TabM	7.9234 ± 0.0045	7.8584 ± 0.0132
TabICL	7.9457 ± 0.0025	7.9072 ± 0.0137
AutoPyTorch	7.8215	
AutoKeras	7.6977	

tabarena/customer_satisfaction_in_airline ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.9408 ± 0.0007	9.9488 ± 0.0008
RealMLP	9.9511 ± 0.0004	9.9518 ± 0.0007
LightGBM	9.9415 ± 0.0004	9.9437 ± 0.0003
TabM	9.9518 ± 0.0006	9.9535 ± 0.0005
TabICL	9.9475 ± 0.0003	9.9476 ± 0.0004
AutoPyTorch	9.9261	
AutoKeras	9.8125	

tabarena/Diabetes130US ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	6.6634 ± 0.0100	6.6510 ± 0.0317
RealMLP	6.7708 ± 0.0312	6.5302 ± 0.3399
LightGBM	6.6224 ± 0.0240	6.6612 ± 0.0247
TabM	6.6973 ± 0.0296	6.7207 ± 0.0234
TabICL	6.6433 ± 0.0999	6.4517 ± 0.0410
AutoPyTorch	6.3983	
AutoKeras	6.0083	

tabarena/Food_Delivery_Time ↓

Model	Default HPO	Agentic HPO
MLP	7.6184 ± 0.0065	7.3969 ± 0.0129
RealMLP	7.4429 ± 0.0144	7.4876 ± 0.0108
LightGBM	7.5184 ± 0.0024	7.2845 ± 0.0011
TabM	7.7018 ± 0.0174	7.3772 ± 0.0059
TabICL	7.6786 ± 0.0226	7.4968 ± 0.0000
AutoPyTorch	7.7677	
AutoKeras	8.0159	

tabarena/healthcare_insurance_expenses ($\times 10^{3}$) ↓

Model	Default HPO	Agentic HPO
MLP	4.3154 ± 0.0611	4.3432 ± 0.0585
RealMLP	4.2618 ± 0.0441	4.1889 ± 0.0545
LightGBM	4.2711 ± 0.0252	4.3164 ± 0.1070
TabM	4.1117 ± 0.0085	4.1942 ± 0.0428
TabICL	4.0512 ± 0.0046	4.0887 ± 0.0328
AutoPyTorch	4.6827	
AutoKeras	4.3969	

tabarena/HR_Analytics_Job_Change_of_Data_Scientists ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.0911 ± 0.0111	8.0883 ± 0.0096
RealMLP	8.1093 ± 0.0160	8.1034 ± 0.0218
LightGBM	8.0685 ± 0.0134	8.0531 ± 0.0221
TabM	8.1173 ± 0.0068	8.1175 ± 0.0060
TabICL	8.0661 ± 0.0627	8.1450 ± 0.0100
AutoPyTorch	8.0030	
AutoKeras	7.9288	

tabarena/diabetes ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.1096 ± 0.0801	8.2129 ± 0.1661
RealMLP	8.0019 ± 0.2015	8.2506 ± 0.1144
LightGBM	8.0315 ± 0.1222	7.9920 ± 0.1538
TabM	7.9481 ± 0.0844	8.3092 ± 0.0408
TabICL	8.4487 ± 0.0036	8.4329 ± 0.0102
AutoPyTorch	8.0877	
AutoKeras	8.1533	

tabarena/E-CommereShippingData ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	7.4061 ± 0.0156	7.3510 ± 0.0548
RealMLP	7.4130 ± 0.0404	7.4087 ± 0.0355
LightGBM	7.4183 ± 0.0089	7.3755 ± 0.0321
TabM	7.4042 ± 0.0242	7.3975 ± 0.0193
TabICL	7.4065 ± 0.0113	7.3961 ± 0.0144
AutoPyTorch	7.3723	
AutoKeras	7.3439	

tabarena/GiveMeSomeCredit ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.7016 ± 0.0023	8.7004 ± 0.0023
RealMLP	8.7012 ± 0.0038	8.6886 ± 0.0042
LightGBM	8.6948 ± 0.0012	8.6812 ± 0.0014
TabM	8.6901 ± 0.0020	8.6940 ± 0.0035
TabICL	8.6447 ± 0.1262	8.6993 ± 0.0023
AutoPyTorch	8.5903	
AutoKeras	8.3068	

tabarena/heloc ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	7.9526 ± 0.0074	7.9523 ± 0.0056
RealMLP	7.9769 ± 0.0021	7.9419 ± 0.0180
LightGBM	7.9061 ± 0.0186	7.9584 ± 0.0066
TabM	7.9618 ± 0.0059	7.9752 ± 0.0056
TabICL	8.0047 ± 0.0072	8.0102 ± 0.0042
AutoPyTorch	7.8953	
AutoKeras	7.8187	

tabarena/in_vehicle_coupon_recommendation ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.2448 ± 0.0308	8.2877 ± 0.0258
RealMLP	8.4424 ± 0.0155	8.4188 ± 0.0199
LightGBM	8.4402 ± 0.0067	8.4756 ± 0.0056
TabM	8.5234 ± 0.0183	8.5315 ± 0.0154
TabICL	8.4973 ± 0.0291	8.4963 ± 0.0020
AutoPyTorch	8.2578	
AutoKeras	7.3155	

tabarena/jm1 ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	7.3112 ± 0.0433	7.3877 ± 0.0272
RealMLP	7.4373 ± 0.0399	7.4161 ± 0.0271
LightGBM	7.5327 ± 0.0232	7.4334 ± 0.0633
TabM	7.3276 ± 0.0180	7.3322 ± 0.0387
TabICL	7.4488 ± 0.1107	7.7270 ± 0.0740
AutoPyTorch	7.3836	
AutoKeras	7.3327	

tabarena/kddcup09_appetency ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.1465 ± 0.0365	8.0903 ± 0.0258
RealMLP	8.1760 ± 0.0284	7.6745 ± 0.5382
LightGBM	8.0864 ± 0.0303	8.1256 ± 0.0408
TabM	8.1329 ± 0.0259	8.1542 ± 0.0302
TabICL	8.1179 ± 0.0126	8.1086 ± 0.0134
AutoPyTorch	6.2612	
AutoKeras	7.5823	

tabarena/miami_housing ($\times 10^{4}$) ↓

Model	Default HPO	Agentic HPO
MLP	8.8229 ± 0.0985	8.6732 ± 0.1253
RealMLP	8.4009 ± 0.1724	8.3701 ± 0.0671
LightGBM	9.1026 ± 0.0869	9.0188 ± 0.0621
TabM	8.2868 ± 0.0378	8.3488 ± 0.1302
TabICL	7.9680 ± 0.0340	7.9111 ± 0.0163
AutoPyTorch	9.2342	
AutoKeras	9.0830	

tabarena/NATICUSdroid ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.8505 ± 0.0033	9.8331 ± 0.0107
RealMLP	9.8443 ± 0.0050	9.8460 ± 0.0055
LightGBM	9.8538 ± 0.0013	9.8442 ± 0.0026
TabM	9.8509 ± 0.0019	9.8461 ± 0.0055
TabICL	9.8665 ± 0.0099	9.8580 ± 0.0148
AutoPyTorch	9.8165	
AutoKeras	9.8083	

tabarena/online_shoppers_intention ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.2454 ± 0.0153	9.2592 ± 0.0099
RealMLP	9.2977 ± 0.0135	9.2838 ± 0.0078
LightGBM	9.2484 ± 0.0235	9.2670 ± 0.0036
TabM	9.2785 ± 0.0121	9.2689 ± 0.0051
TabICL	9.3245 ± 0.0117	9.3004 ± 0.0157
AutoPyTorch	9.2502	
AutoKeras	8.9242	

tabarena/physiochemical_protein ↓

Model	Default HPO	Agentic HPO
MLP	3.5443 ± 0.0155	3.6026 ± 0.0139
RealMLP	3.2988 ± 0.0134	3.2711 ± 0.0112
LightGBM	3.5027 ± 0.0022	3.2991 ± 0.0044
TabM	3.3609 ± 0.0255	3.3444 ± 0.0138
TabICL	3.0133 ± 0.0066	3.0075 ± 0.0040
AutoPyTorch	3.6083	
AutoKeras	3.8070	

tabarena/polish_companies_bankruptcy ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.2275 ± 0.0604	9.3093 ± 0.0684
RealMLP	9.3074 ± 0.0902	9.5427 ± 0.0418
LightGBM	9.5801 ± 0.0271	9.6194 ± 0.0178
TabM	9.5876 ± 0.0238	9.2369 ± 0.0555
TabICL	9.8386 ± 0.0074	9.8576 ± 0.0073
AutoPyTorch	9.3865	
AutoKeras	8.6602	

tabarena/qsar-biodeg ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.1409 ± 0.0369	9.1758 ± 0.1390
RealMLP	9.1275 ± 0.0500	9.2730 ± 0.0261
LightGBM	9.0573 ± 0.0837	9.0792 ± 0.0891
TabM	9.3032 ± 0.0186	9.2689 ± 0.0367
TabICL	9.3723 ± 0.0072	9.3363 ± 0.0088
AutoPyTorch	8.9202	
AutoKeras	9.2520	

tabarena/QSAR-TID-11 ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	8.8851 ± 0.0623	8.3229 ± 0.0459
RealMLP	8.4800 ± 0.0718	8.5389 ± 0.0369
LightGBM	8.3262 ± 0.0274	8.2736 ± 0.0224
TabM	8.4739 ± 0.0186	8.2702 ± 0.0289
TabICL	8.7313 ± 0.0534	8.7626 ± 0.0475
AutoPyTorch	8.8098	
AutoKeras	9.1580	

tabarena/QSAR_fish_toxicity ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	9.8934 ± 0.3439	9.7175 ± 0.1754
RealMLP	9.2329 ± 0.0780	9.4535 ± 0.1564
LightGBM	9.6557 ± 0.1513	9.6247 ± 0.1816
TabM	9.3465 ± 0.0941	9.4143 ± 0.0778
TabICL	8.9515 ± 0.0132	8.9564 ± 0.0000
AutoPyTorch	9.5082	
AutoKeras	9.8908	

tabarena/splice ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	1.1486 ± 0.0190	0.9329 ± 0.0475
RealMLP	1.0274 ± 0.0459	1.0824 ± 0.5004
LightGBM	1.0272 ± 0.0422	0.9034 ± 0.0424
TabM	0.9576 ± 0.0111	1.0152 ± 0.0406
TabICL	0.7346 ± 0.0099	0.7314 ± 0.0058
AutoPyTorch	1.4024	
AutoKeras	1.9949	

tabarena/taiwanese_bankruptcy_prediction ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.3664 ± 0.0071	8.9853 ± 0.1144
RealMLP	9.2386 ± 0.1752	9.2692 ± 0.2576
LightGBM	9.4283 ± 0.0450	9.3512 ± 0.1165
TabM	9.4117 ± 0.0177	9.4787 ± 0.0163
TabICL	9.4728 ± 0.0077	9.5173 ± 0.0067
AutoPyTorch	9.1591	
AutoKeras	8.7867	

tabarena/wine_quality ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	6.7464 ± 0.0426	6.5243 ± 0.0594
RealMLP	6.0529 ± 0.0300	6.0895 ± 0.0278
LightGBM	6.1508 ± 0.0125	6.0856 ± 0.0196
TabM	6.3776 ± 0.0724	6.2001 ± 0.0247
TabICL	5.8641 ± 0.0064	5.8630 ± 0.0047
AutoPyTorch	6.6550	
AutoKeras	6.6474	

tabred/cooking-time ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	4.8084 ± 0.0041	4.8119 ± 0.0025
RealMLP	4.8131 ± 0.0055	4.8035 ± 0.0029
LightGBM	4.8248 ± 0.0009	4.8235 ± 0.0008
TabM	4.8029 ± 0.0037	4.8022 ± 0.0033
TabICL	4.8954 ± 0.0056	4.8828 ± 0.0059
AutoPyTorch	4.8263	
AutoKeras	4.9050	

tabred/delivery-eta ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	5.5289 ± 0.0164	5.5336 ± 0.0197
RealMLP	5.4760 ± 0.0096	5.4708 ± 0.0102
LightGBM	5.4694 ± 0.0016	5.4654 ± 0.0010
TabM	5.5013 ± 0.0167	5.4949 ± 0.0092
TabICL	5.5830 ± 0.0061	5.5812 ± 0.0047
AutoPyTorch	5.5704	
AutoKeras	5.5888	

tabred/ecom-offers ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	5.9667 ± 0.0181	6.0484 ± 0.0138
RealMLP	5.9346 ± 0.0202	5.8116 ± 0.1743
LightGBM	5.7377 ± 0.0326	5.7700 ± 0.0565
TabM	5.8804 ± 0.0146	6.0097 ± 0.0137
TabICL	5.8372 ± 0.1184	5.9924 ± 0.0298
AutoPyTorch	6.1761	
AutoKeras	5.8387	

tabred/homecredit-default ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	8.5875 ± 0.0137	8.5842 ± 0.0103
RealMLP	8.6299 ± 0.0130	8.6260 ± 0.0051
LightGBM	8.6790 ± 0.0033	8.6757 ± 0.0031
TabM	8.6438 ± 0.0071	8.6353 ± 0.0046
TabICL	8.0993 ± 0.0194	8.0900 ± 0.0161
AutoPyTorch	6.3470	
AutoKeras	5.8340	

tabred/homesite-insurance ($\times 10^{-1}$) ↑

Model	Default HPO	Agentic HPO
MLP	9.5994 ± 0.0077	9.6137 ± 0.0052
RealMLP	9.6511 ± 0.0032	9.6329 ± 0.0046
LightGBM	9.6063 ± 0.0017	9.6111 ± 0.0029
TabM	9.6246 ± 0.0034	9.6297 ± 0.0017
TabICL	9.4678 ± 0.0351	9.5267 ± 0.0076
AutoPyTorch	9.4293	
AutoKeras	9.3546	

tabred/maps-routing ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	1.6176 ± 0.0018	1.6207 ± 0.0020
RealMLP	1.6117 ± 0.0021	1.6089 ± 0.0015
LightGBM	1.6165 ± 0.0004	1.6166 ± 0.0004
TabM	1.6100 ± 0.0012	1.6101 ± 0.0008
TabICL	1.6886 ± 0.0026	1.6897 ± 0.0016
AutoPyTorch	1.6537	
AutoKeras	1.7349	

tabred/sberbank-housing ($\times 10^{-1}$) ↓

Model	Default HPO	Agentic HPO
MLP	2.4056 ± 0.0251	2.3904 ± 0.0285
RealMLP	2.3101 ± 0.0090	2.3578 ± 0.0232
LightGBM	2.4882 ± 0.0154	2.5381 ± 0.0208
TabM	2.3646 ± 0.0235	2.3863 ± 0.0174
TabICL	2.8047 ± 0.0498	2.4961 ± 0.0145
AutoPyTorch	3.6158	
AutoKeras	5.4295	

tabred/weather ↓

Model	Default HPO	Agentic HPO
MLP	1.5212 ± 0.0032	1.5136 ± 0.0021
RealMLP	1.4363 ± 0.0020	1.4389 ± 0.0016
LightGBM	1.4619 ± 0.0009	1.4657 ± 0.0009
TabM	1.4669 ± 0.0025	1.4654 ± 0.0021
TabICL	1.5829 ± 0.0045	1.5670 ± 0.0024
AutoPyTorch	1.5236	
AutoKeras	1.6028	

E.3 ENSEMBLE RESULTS

Table 23 reports the per-dataset greedy-ensemble results behind the ensembling discussion in Section 4.1. Each subtable lists the four families (MLP†, TabM†, RealMLP, and LightGBM) with their base and agentic search spaces, and for each (family, space) reports three quantities: the ensemble's test metric (*Ens.*), the test metric of the single strongest ensemble member (*Best*), and the mean pairwise test-prediction correlation across the selected members (ρ; lower $\rightarrow$ more diverse pool). Ensembles are constructed by greedy forward selection following Caruana et al. (2004) on a pool of 100 (20 for the large datasets) random-hyperparameter configurations per (family, space). In the few cases where greedy selection collapses to a single distinct member, the pairwise correlation is undefined, which we denote by "–" in the table. Below the family rows, each subtable lists the ensemble rows of the AutoPytorch and AutoGluon (see Appendix D.1 for details). We include the default AutoGluon portfolio under *Default* column, and AutoGluon with agentic RealMLP and TabM† under *Agentic*. Arrows indicate whether higher or lower values are better.

Table 23: Per-dataset greedy-ensemble results for each of the 45 datasets. Each subtable reports, for the default and agentic search spaces of every model family, the ensemble test metric (Ens.), the test metric of its strongest single member (Best), and the mean pairwise prediction correlation among selected members (ρ). Arrows indicate whether higher or lower values are better.

black-friday ($\times 10^3$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	3.4609	3.4762	0.980	3.4595	3.4787	0.963
TabM†	3.4565	3.4602	0.983	3.4599	3.4645	0.966
LightGBM	3.4538	3.4558	0.982	3.4540	3.4645	0.947
RealMLP	3.4458	3.4541	0.992	3.4432	3.4471	0.976
AutoPyTorch	3.4638	–	–	–	–	–
AutoGluon	3.4432	–	–	3.4417	–	–

california ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	4.4171	4.5278	0.980	4.3439	4.5809	0.947
TabM†	4.1570	4.2247	0.985	4.0790	4.2081	0.980
LightGBM	4.2722	4.3053	0.967	4.3162	4.4208	0.978
RealMLP	4.0541	4.1375	0.983	3.9219	3.9471	0.982
AutoPyTorch	4.2516	–	–	–	–	–
AutoGluon	4.0306	–	–	3.9467	–	–

churn ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.6100	8.5750	0.951	8.6300	8.6500	0.884
TabM†	8.5750	8.5700	0.867	8.5800	8.5850	0.856
LightGBM	8.6100	8.5750	0.960	8.6250	8.5600	0.409
RealMLP	8.6200	8.5600	0.986	8.6100	8.6250	0.984
AutoPyTorch	8.6000	–	–	–	–	–
AutoGluon	8.6050	–	–	8.6550	–	–

diamond ($\times 10^2$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	5.1618	5.2748	0.999	5.1070	5.1981	0.997
TabM†	5.1751	5.2088	0.999	5.1541	5.1925	0.998
LightGBM	5.4697	5.4922	0.997	5.2886	5.4848	0.996
RealMLP	5.1536	5.2238	0.999	5.1586	5.2889	0.999
AutoPyTorch	5.1642	–	–	–	–	–
AutoGluon	5.0869	–	–	5.0497	–	–

house ($\times 10^4$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	3.0591	3.2206	0.959	3.0086	3.1091	0.920
TabM†	3.0332	3.0416	0.978	2.9927	3.0683	0.795
LightGBM	3.1224	3.1500	0.948	3.1270	3.1635	0.938
RealMLP	3.0202	3.0656	0.968	2.9877	3.0371	0.965
AutoPyTorch	3.0247	–	–	–	–	–
AutoGluon	2.9352	–	–	2.9503	–	–

microsoft ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.4381	7.4595	0.943	7.4458	7.4930	0.929
TabM†	7.4217	7.4316	0.983	7.4258	7.4535	0.941
LightGBM	7.4217	7.4272	0.957	7.4346	7.4371	0.981
RealMLP	7.4413	7.4725	0.947	7.4248	7.4382	0.943
AutoPyTorch	7.4313	–	–	–	–	–
AutoGluon	7.3968	–	–	7.3998	–	–

tabarena/airfoil_self_noise ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	1.0921	1.0719	0.992	1.0495	1.2694	0.987
TabM†	1.0417	1.0543	0.997	0.9682	1.0703	0.996
LightGBM	1.6305	1.5509	0.977	1.6208	1.8714	0.924
RealMLP	1.0711	1.0425	0.995	1.0546	1.1026	0.993
AutoPyTorch	1.4413	–	–	–	–	–
AutoGluon	1.1591	–	–	1.0240	–	–

tabarena/Amazon_employee_access ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.4789	8.3096	0.973	8.5700	8.2742	0.882
TabM†	8.3749	8.2706	0.919	8.4651	8.5256	-0.106
LightGBM	8.3321	8.1961	0.974	8.5580	8.4597	0.957
RealMLP	8.6927	8.5656	0.983	8.5490	8.4874	0.953
AutoPyTorch	8.2391	–	–	–	–	–
AutoGluon	8.7546	–	–	8.7749	–	–

tabarena/Another-Dataset-on-used-Fiat-500 ($\times 10^2$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.2749	7.4761	0.992	7.1394	7.1956	0.978
TabM†	7.2183	7.2410	0.997	7.2735	7.2945	0.970
LightGBM	7.2233	7.1797	0.965	7.1508	7.0806	0.970
RealMLP	6.9871	7.1376	0.988	7.2087	7.5569	0.984
AutoPyTorch	7.0074	–	–	–	–	–
AutoGluon	6.9228	–	–	6.9843	–	–

tabarena/APSFailure ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.9388	9.9243	0.996	9.9531	9.8932	0.986
TabM†	9.9400	9.9317	0.996	9.9543	9.9250	0.992
LightGBM	9.8816	9.9071	0.983	9.9428	9.9421	0.982
RealMLP	9.9254	9.9397	0.997	9.9404	9.9328	0.996
AutoPyTorch	9.9338	–	–	–	–	–
AutoGluon	9.9617	–	–	9.9509	–	–

tabarena/bank-marketing ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.7486	7.7187	0.996	7.7263	7.6759	0.939
TabM†	7.7408	7.7406	0.998	7.7250	7.7256	0.994
LightGBM	7.6496	7.5439	0.958	7.6322	7.5159	0.976
RealMLP	7.7214	7.6891	0.995	7.7328	7.7042	0.994
AutoPyTorch	7.7188	–	–	–	–	–
AutoGluon	7.7255	–	–	7.7536	–	–

tabarena/Bioresponse ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.6307	8.5387	0.866	8.7010	8.6130	0.774
TabM†	8.7630	8.7514	0.968	8.7947	8.6292	0.349
LightGBM	8.8308	8.7361	0.668	8.8487	8.8535	0.495
RealMLP	8.7836	8.5722	0.878	8.7568	8.6600	0.793
AutoPyTorch	8.7056	–	–	–	–	–
AutoGluon	8.8883	–	–	8.8907	–	–

tabarena/concrete_compressive_strength ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	4.5909	4.6376	0.989	4.1806	4.1483	0.826
TabM†	4.4148	4.6771	0.984	4.4071	4.7297	0.978
LightGBM	4.6905	4.6727	0.978	4.4771	4.5654	0.972
RealMLP	4.6129	4.9984	0.987	4.4865	4.9832	0.707
AutoPyTorch	4.5269	–	–	–	–	–
AutoGluon	4.1461	–	–	3.9968	–	–

tabarena/credit_card_clients_default ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.8881	7.8898	0.991	7.8919	7.8699	0.844
TabM†	7.9085	7.9051	0.998	7.9266	7.8867	0.982
LightGBM	7.8757	7.8719	0.931	7.8643	7.8703	0.792
RealMLP	7.8777	7.8919	0.977	7.9084	7.9152	0.979
AutoPyTorch	7.8629	–	–	–	–	–
AutoGluon	7.9133	–	–	7.9356	–	–

tabarena/customer_satisfaction_in_airline ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.9456	9.9414	0.986	9.9462	9.9381	0.974
TabM†	9.9519	9.9512	0.997	9.9527	9.9514	0.459
LightGBM	9.9418	9.9420	0.832	9.9425	9.9425	0.991
RealMLP	9.9507	9.9479	0.993	9.9517	9.9496	0.991
AutoPyTorch	9.9444	–	–	–	–	–
AutoGluon	9.9553	–	–	9.9547	–	–

tabarena/Diabetes130US ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	6.6519	6.6647	0.998	6.7481	6.6587	0.986
TabM†	6.7300	6.7372	1.000	6.7645	6.6671	0.997
LightGBM	6.6741	6.6575	0.989	6.7327	6.6237	0.921
RealMLP	6.7327	6.7111	0.990	6.7642	6.7286	0.979
AutoPyTorch	6.7263	–	–	–	–	–
AutoGluon	6.8475	–	–	6.8117	–	–

tabarena/Food_Delivery_Time ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.6165	7.6147	0.936	7.4622	7.4826	0.935
TabM†	7.6896	7.6884	0.944	7.4060	7.4684	0.937
LightGBM	7.5217	7.5367	0.899	7.3161	7.3560	0.856
RealMLP	7.3778	7.3825	0.951	7.5389	7.6036	0.924
AutoPyTorch	7.4385	–	–	–	–	–
AutoGluon	7.4264	–	–	7.5950	–	–

tabarena/healthcare_insurance_expenses ($\times 10^{3}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	4.2428	4.4211	0.992	4.2734	4.6018	0.988
TabM†	4.1157	4.1540	0.995	4.2239	4.2813	0.992
LightGBM	4.2900	4.2384	0.990	4.3421	4.3323	0.964
RealMLP	4.1561	4.1509	0.994	4.1224	4.1451	0.993
AutoPyTorch	4.3118	–	–	–	–	–
AutoGluon	4.0551	–	–	4.0598	–	–

tabarena/HR_Analytics_Job_Change_of_Data_Scientists ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.1123	8.0956	0.986	8.1218	8.0863	0.620
TabM†	8.1293	8.1237	0.993	8.1067	8.0417	0.981
LightGBM	8.0766	8.0706	0.854	8.0970	8.1152	0.786
RealMLP	8.1128	8.1023	0.987	8.1219	8.0735	0.984
AutoPyTorch	8.0876	–	–	–	–	–
AutoGluon	8.1381	–	–	8.1415	–	–

tabarena/diabetes ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.2550	8.1754	0.783	8.3527	8.4016	0.585
TabM†	7.5817	7.5783	0.986	8.2483	8.2584	0.743
LightGBM	8.1991	7.9973	0.999	7.9227	7.2828	1.000
RealMLP	8.4357	8.4083	0.920	8.2811	8.3447	0.919
AutoPyTorch	8.1566	–	–	–	–	–
AutoGluon	8.4076	–	–	8.3835	–	–

tabarena/E-CommereShippingData ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.3964	7.4121	0.939	7.3701	7.3701	–
TabM†	7.3678	7.3547	0.995	7.3983	7.4361	0.502
LightGBM	7.3731	7.3401	0.694	7.4145	7.4145	-0.012
RealMLP	7.4131	7.3659	0.971	7.4248	7.4181	0.939
AutoPyTorch	7.4104	–	–	–	–	–
AutoGluon	7.4040	–	–	7.4084	–	–

tabarena/GiveMeSomeCredit ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.7073	8.6997	0.999	8.7016	8.6879	0.988
TabM†	8.7062	8.6994	0.999	8.7122	8.7081	0.994
LightGBM	8.6910	8.6881	0.955	8.6628	8.6756	0.740
RealMLP	8.6931	8.6834	0.999	8.6925	8.6911	0.985
AutoPyTorch	8.6934	–	–	–	–	–
AutoGluon	8.7078	–	–	8.7084	–	–

tabarena/heloc ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.9570	7.9454	0.974	7.9828	7.9527	0.394
TabM†	7.9660	7.9696	0.992	7.9823	7.9663	0.526
LightGBM	7.9473	7.9340	0.743	7.9429	7.9444	0.036
RealMLP	7.9982	7.9770	0.954	7.9889	7.9830	0.950
AutoPyTorch	7.9512	–	–	–	–	–
AutoGluon	7.9772	–	–	7.9812	–	–

tabarena/in_vehicle_coupon_recommendation ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.3507	8.2174	0.794	8.4306	8.3132	0.743
TabM†	8.5539	8.5323	0.905	8.5120	8.5156	0.530
LightGBM	8.4024	8.4187	0.610	8.3935	8.3868	-0.319
RealMLP	8.4935	8.4211	0.927	8.5175	8.4445	0.885
AutoPyTorch	8.4038	–	–	–	–	–
AutoGluon	8.5373	–	–	8.5634	–	–

tabarena/jm1 ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	7.3901	7.2929	0.977	7.4216	7.3109	0.913
TabM†	7.4311	7.3634	0.986	7.5160	7.3839	0.979
LightGBM	7.5870	7.5646	0.931	7.6147	7.4196	0.642
RealMLP	7.6450	7.4614	0.958	7.6531	7.4825	0.943
AutoPyTorch	7.7106	–	–	–	–	–
AutoGluon	7.5190	–	–	7.6760	–	–

tabarena/kddcup09_appetency ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.1603	8.1518	0.999	8.0949	8.0878	0.770
TabM†	8.1484	8.1584	1.000	8.1658	8.1952	0.995
LightGBM	7.9505	8.1172	0.999	7.1522	7.1522	0.015
RealMLP	8.1288	8.0405	1.000	8.2018	8.1551	0.996
AutoPyTorch	7.8779	–	–	–	–	–
AutoGluon	8.4368	–	–	8.3163	–	–

tabarena/miami_housing ($\times 10^{4}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.5496	8.9288	0.987	8.3634	8.7704	0.978
TabM†	8.5352	8.8032	0.989	8.1855	8.3433	0.988
LightGBM	9.0131	8.8094	0.976	9.0017	8.9335	0.931
RealMLP	8.3397	8.3321	0.993	8.0782	8.4177	0.990
AutoPyTorch	8.3606	–	–	–	–	–
AutoGluon	7.8451	–	–	7.8286	–	–

tabarena/NATICUSdroid ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.8473	9.8325	0.977	9.8572	9.8495	0.909
TabM†	9.8532	9.8504	0.997	9.8653	9.8522	0.977
LightGBM	9.8440	9.8454	0.736	9.8463	9.8487	0.636
RealMLP	9.8562	9.8485	0.990	9.8604	9.8520	0.983
AutoPyTorch	9.8457	–	–	–	–	–
AutoGluon	9.8646	–	–	9.8668	–	–

tabarena/online_shoppers_intention ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.2681	9.2540	0.992	9.2108	9.2693	0.724
TabM†	9.2869	9.2832	0.997	9.2768	9.2848	0.929
LightGBM	9.2431	9.2589	0.914	9.2372	9.2628	0.869
RealMLP	9.2790	9.2205	0.986	9.2969	9.2815	0.977
AutoPyTorch	9.2705	–	–	–	–	–
AutoGluon	9.3096	–	–	9.3084	–	–

tabarena/physiochemical_protein ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	3.3828	3.5599	0.910	3.3040	3.4991	0.898
TabM†	3.3422	3.4287	0.969	3.2783	3.3622	0.950
LightGBM	3.7260	3.7472	0.846	3.3345	3.4489	0.946
RealMLP	3.1742	3.2926	0.942	3.1772	3.2847	0.940
AutoPyTorch	3.3437	–	–	–	–	–
AutoGluon	3.2008	–	–	3.1264	–	–

tabarena/polish_companies_bankruptcy ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.3699	9.3477	0.970	9.6607	9.4425	0.924
TabM†	9.6205	9.5664	0.979	9.5262	9.5888	0.701
LightGBM	9.5611	9.5821	0.940	9.6257	9.6295	0.958
RealMLP	9.6899	9.4042	0.975	9.6370	9.2893	0.974
AutoPyTorch	9.6560	–	–	–	–	–
AutoGluon	9.7467	–	–	9.7515	–	–

tabarena/qsar-biodeg ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.2545	9.2069	0.900	9.3176	9.2275	0.849
TabM†	9.2985	9.1950	0.976	9.3721	9.3753	0.749
LightGBM	9.1896	9.1896	0.463	9.0525	9.0702	0.745
RealMLP	9.2585	9.2138	0.923	9.3155	9.0901	0.910
AutoPyTorch	9.1445	–	–	–	–	–
AutoGluon	9.3068	–	–	9.2729	–	–

tabarena/QSAR-TID-11 ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.5913	8.7609	0.954	8.3543	8.9867	0.929
TabM†	8.4130	8.4003	0.996	8.2702	8.3575	0.978
LightGBM	8.3660	8.3683	0.872	8.4294	8.3663	0.655
RealMLP	8.2842	8.3872	0.982	8.4596	8.4647	0.981
AutoPyTorch	8.2661	–	–	–	–	–
AutoGluon	8.1559	–	–	8.2598	–	–

tabarena/QSAR_fish_toxicity ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	0.9459	0.9685	0.968	0.9204	0.9512	0.877
TabM†	0.9357	0.9392	0.968	0.9389	0.9454	0.953
LightGBM	0.9621	1.0272	0.880	0.9251	0.9791	0.835
RealMLP	0.9099	0.9526	0.913	0.9139	0.9390	0.920
AutoPyTorch	0.9303	–	–	–	–	–
AutoGluon	0.9083	–	–	0.9046	–	–

tabarena/splice ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	1.1686	1.1937	0.959	0.9582	1.1222	0.958
TabM†	1.0282	0.9202	0.998	0.8593	0.8383	0.991
LightGBM	0.9863	1.0077	0.985	0.8797	0.9520	0.989
RealMLP	0.9563	1.0102	0.991	0.8957	0.9865	0.988
AutoPyTorch	1.1290	–	–	–	–	–
AutoGluon	0.7634	–	–	0.8551	–	–

tabarena/taiwanese_bankruptcy_prediction ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.3644	9.3983	0.997	9.4326	9.3093	0.974
TabM†	9.4515	9.4238	0.998	9.4295	9.3433	0.990
LightGBM	9.4484	9.4721	0.989	9.3898	9.3192	0.990
RealMLP	9.4868	9.2300	0.992	9.4259	9.3598	0.989
AutoPyTorch	9.3738	–	–	–	–	–
AutoGluon	9.5032	–	–	9.4773	–	–

tabarena/wine_quality ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	6.4366	6.6292	0.903	6.1237	6.6758	0.722
TabM†	6.1524	6.4087	0.890	6.0114	6.2220	0.838
LightGBM	6.2225	6.1372	0.852	6.2050	6.4665	0.728
RealMLP	5.9686	6.1871	0.895	5.9959	6.1459	0.905
AutoPyTorch	6.0431	–	–	–	–	–
AutoGluon	5.8837	–	–	5.8556	–	–

tabred/cooking-time ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	4.7999	4.8093	0.991	4.8031	4.8344	0.986
TabM†	4.7987	4.8011	0.998	4.7986	4.7982	0.996
LightGBM	4.8194	4.8226	0.990	4.8259	4.8281	0.995
RealMLP	4.8024	4.8119	0.993	4.8048	4.8141	0.987
AutoPyTorch	4.7988	–	–	–	–	–
AutoGluon	4.7882	–	–	4.7891	–	–

tabred/delivery-eta ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	5.5079	5.5270	0.990	5.5025	5.5323	0.981
TabM†	5.4759	5.4800	0.995	5.4652	5.4808	0.988
LightGBM	5.4639	5.4643	0.992	5.5105	5.5358	0.949
RealMLP	5.4652	5.4769	0.990	5.4544	5.4723	0.989
AutoPyTorch	5.4697	–	–	–	–	–
AutoGluon	5.4580	–	–	5.4573	–	–

tabred/ecom-offers ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	5.9635	5.9635	–	5.9944	5.9944	–
TabM†	5.8574	5.8574	–	5.9934	5.9511	0.983
LightGBM	5.8303	5.7756	0.938	5.8483	5.8433	1.000
RealMLP	5.9450	5.9312	0.991	5.9316	5.8410	0.866
AutoPyTorch	6.2650	–	–	–	–	–
AutoGluon	6.3186	–	–	6.2162	–	–

tabred/homecredit-default ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	8.6193	8.5998	0.999	8.6410	8.5737	0.997
TabM†	8.6472	8.6348	0.999	8.6519	8.5944	0.997
LightGBM	8.6786	8.6742	0.997	8.5615	8.5640	0.982
RealMLP	8.6340	8.6020	0.998	8.6474	8.6269	0.992
AutoPyTorch	8.6189	–	–	–	–	–
AutoGluon	8.6221	–	–	8.6620	–	–

tabred/homesite-insurance ($\times 10^{-1}$) ↑

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	9.6024	9.5899	0.984	9.5985	9.5602	0.966
TabM†	9.6230	9.6230	–	9.5707	9.5833	0.982
LightGBM	9.6052	9.5986	0.991	9.6023	9.6057	0.952
RealMLP	9.6394	9.6362	0.995	9.6420	9.6353	0.988
AutoPyTorch	9.6367	–	–	–	–	–
AutoGluon	9.6493	–	–	9.6477	–	–

tabred/maps-routing ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	1.6114	1.6200	0.996	1.6110	1.6277	0.992
TabM†	1.6097	1.6102	0.999	1.6105	1.6149	0.992
LightGBM	1.6162	1.6196	0.998	1.6178	1.6187	0.996
RealMLP	1.6103	1.6131	0.998	1.6065	1.6104	0.996
AutoPyTorch	1.6099	–	–	–	–	–
AutoGluon	1.6006	–	–	1.6061	–	–

tabred/sberbank-housing ($\times 10^{-1}$) ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	2.3853	2.4321	0.969	2.4756	2.4758	0.949
TabM†	2.3388	2.3320	0.986	2.4203	2.5490	0.930
LightGBM	2.5091	2.5018	0.983	2.7775	2.8120	0.856
RealMLP	2.3945	2.3928	0.966	2.2669	2.2862	0.984
AutoPyTorch	2.3370	–	–	–	–	–
AutoGluon	2.2999	–	–	2.2180	–	–

tabred/weather ↓

	Default			Agentic		
Model	Ens.	Best	ρ	Ens.	Best	ρ
MLP†	1.4923	1.5298	0.990	1.4863	1.5259	0.985
TabM†	1.4778	1.4818	0.997	1.4610	1.4836	0.991
LightGBM	1.4592	1.4618	0.994	1.4871	1.4964	0.993
RealMLP	1.4361	1.4471	0.994	1.4328	1.4412	0.993
AutoPyTorch	1.4491	–	–	–	–	–
AutoGluon	1.4368	–	–	1.4441	–	–

F TabArena Per-Dataset Results

Table 24 reports the per-dataset results behind the TabArena-Lite comparison in Appendix A. Each subtable lists the five model families and reports, for the base and the agentic (-A) search space, the score of the default configuration (Def.), of the best of the 200 random configurations by validation score (Tuned), and of the greedy ensemble built from the same configurations (T+E). Scores use the official TabArena metric of each dataset: AUROC for binary classification, log-loss for multiclass, and RMSE for regression. Base rows are the official TabArena results, except MLP†, which has no official TabArena counterpart and whose base row is our own run under the same protocol, sharing its default configuration with MLP†-A. The official TabICLv2 entry is default-only, so its Tuned and T+E cells are marked as missing.

Table 24: Per-dataset TabArena-Lite results for the 51 datasets: default (Def.), tuned (Tuned), and tuned-plus-ensembled (T+E) test scores for the base and agentic search space of each model family. Arrows indicate whether higher or lower values are better.

airfoil_self_noise, RMSE ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	1.4854	1.3876	1.3396	1.4854	1.4854	1.3031
MLP†	1.1705	1.0631	1.0553	1.1705	1.0168	0.9838
RealMLP	1.1262	1.0575	1.0119	1.0640	0.9813	0.9742
TabICLv2	1.0154	–	–	1.0154	1.0083	0.9914
TabM	1.0875	0.9788	0.9779	1.0959	0.9176	0.9202

Amazon_employee_access, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.3660	8.4437	8.4864	8.3660	8.4917	8.5673
MLP†	8.5315	8.5067	8.5799	8.5315	8.6330	8.6758
RealMLP	8.4428	8.6597	8.7012	8.6365	8.6145	8.7024
TabICLv2	8.5046	–	–	8.5053	8.5012	8.5119
TabM	8.3801	8.4179	8.4909	8.3482	8.7017	8.6747

anneal, Log-loss ($\times 10^{-2}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.2584	4.3466	4.4405	7.2584	5.5082	5.0269
MLP†	6.5717	6.4079	4.4412	6.5717	6.6106	4.1734
RealMLP	3.1609	2.9963	2.9990	3.5202	2.4415	2.4415
TabICLv2	3.2078	–	–	3.1899	2.3954	2.3954
TabM	4.2733	2.6882	2.7093	4.0952	4.7192	4.2572

Another-Dataset-on-used-Fiat-500, RMSE ($\times 10^{2}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.2710	7.0930	6.9478	7.2710	7.2790	7.0273
MLP†	7.3971	7.1931	7.1651	7.3971	7.1698	7.0432
RealMLP	7.1131	7.0236	6.9027	7.1540	7.0659	6.8551
TabICLv2	6.8606	–	–	6.8576	6.8654	6.8516
TabM	7.2404	7.2139	7.1835	7.2313	7.2243	7.0283

APSFailure, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.9464	9.9487	9.9483	9.9464	9.9438	9.9537
MLP†	9.9346	9.9351	9.9418	9.9346	9.9416	9.9538
RealMLP	9.9243	9.9307	9.9481	9.9250	9.9412	9.9571
TabICLv2	9.9534	–	–	9.9546	9.9546	9.9630
TabM	9.9436	9.9445	9.9484	9.9414	9.9390	9.9515

bank-marketing, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.7027	7.7387	7.7417	7.7027	7.7481	7.7517
MLP†	7.7248	7.7366	7.7541	7.7248	7.7495	7.7510
RealMLP	7.7173	7.7328	7.7446	7.7084	7.7571	7.7538
TabICLv2	7.7689	–	–	7.7671	7.7517	7.7276
TabM	7.7437	7.7377	7.7492	7.7368	7.7353	7.7524

Bank_Customer_Churn, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.6936	8.7443	8.7426	8.6936	8.7357	8.7578
MLP†	8.7067	8.7385	8.7320	8.7067	8.7164	8.7559
RealMLP	8.7231	8.7775	8.7808	8.7425	8.7621	8.7841
TabICLv2	8.7842	–	–	8.7836	8.7741	8.7773
TabM	8.7868	8.7887	8.7868	8.7848	8.7816	8.7843

Bioresponse, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.8880	8.8787	8.9029	8.8880	8.8720	8.8873
MLP†	8.5610	8.6478	8.6933	8.5610	8.7718	8.8398
RealMLP	8.7228	8.8186	8.8416	8.7706	8.7706	8.8502
TabICLv2	8.7160	–	–	8.7172	8.6908	8.7482
TabM	8.6549	8.7700	8.7720	8.6518	8.7812	8.8593

blood-transfusion-service-center, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	6.6294	7.1579	7.2193	6.6294	7.1228	7.1351
MLP†	7.1360	7.0877	7.0728	7.1360	6.9605	7.3895
RealMLP	7.4500	7.2482	7.2982	7.1969	7.3237	7.2044
TabICLv2	7.4518	–	–	7.4474	7.4404	7.4544
TabM	7.0272	7.0298	7.0430	6.9605	7.2404	7.1711

churn, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.2870	9.3605	9.3433	9.2870	9.3315	9.3512
MLP†	9.2878	9.2578	9.3075	9.2878	9.3213	9.3136
RealMLP	9.2936	9.3822	9.3791	9.3738	9.3841	9.3855
TabICLv2	9.4502	–	–	9.4475	9.3896	9.4196
TabM	9.3826	9.3735	9.3568	9.3766	9.4102	9.3945

coil2000_insurance_policies, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.4795	7.4791	7.4803	7.4795	7.3837	7.5055
MLP†	7.3933	7.3825	7.4194	7.3933	7.4163	7.4432
RealMLP	7.4531	7.5463	7.5544	7.4695	7.3916	7.5301
TabICLv2	7.6316	–	–	7.6312	7.6391	7.5969
TabM	7.4922	7.4749	7.5207	7.4984	7.3979	7.4428

concrete_compressive_strength, RMSE ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	4.6100	4.3673	4.2318	4.6100	4.2368	4.2096
MLP†	4.6863	4.7111	4.5587	4.6863	3.9086	3.9879
RealMLP	5.0135	4.1444	4.1414	4.8796	4.1396	3.8891
TabICLv2	3.6729	–	–	3.6735	3.6578	3.6594
TabM	4.5057	4.2300	4.3578	4.5182	4.3020	4.1273

credit-g, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.6970	7.7778	7.7974	7.6970	7.6910	7.8026
MLP†	7.6902	7.5590	7.7278	7.6902	7.7026	7.8269
RealMLP	7.8962	7.9671	7.9252	7.8231	7.8803	7.9265
TabICLv2	7.8927	–	–	7.8923	7.8774	7.8996
TabM	7.8726	7.8406	7.8429	7.8782	7.8282	7.8658

credit_card_clients_default, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.8890	7.9021	7.9060	7.8890	7.9335	7.9392
MLP†	7.8785	7.8997	7.9034	7.8785	7.8975	7.9093
RealMLP	7.8979	7.9021	7.9056	7.9028	7.9277	7.9191
TabICLv2	7.9501	–	–	7.9496	7.9431	7.9434
TabM	7.8987	7.9308	7.9302	7.9019	7.9225	7.9264

customer_satisfaction_in_airline, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.9426	9.9446	9.9461	9.9426	9.9426	9.9431
MLP†	9.9405	9.9455	9.9474	9.9405	9.9445	9.9490
RealMLP	9.9485	9.9518	9.9523	9.9459	9.9531	9.9540
TabICLv2	9.9521	–	–	9.9521	9.9521	9.9520
TabM	9.9489	9.9540	9.9539	9.9489	9.9516	9.9529

diabetes, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.1740	8.3534	8.3454	8.1740	8.3628	8.3568
MLP†	8.2671	8.2175	8.2758	8.2671	8.2932	8.3541
RealMLP	8.5007	8.3902	8.4592	8.3728	8.4819	8.4257
TabICLv2	8.4518	–	–	8.4511	8.4284	8.4304
TabM	8.1513	8.2276	8.2195	8.1506	8.1131	8.2523

Diabetes130US, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	6.5737	6.7789	6.8193	6.5737	6.1191	6.2801
MLP†	6.6850	6.7007	6.6983	6.6850	6.7349	6.7601
RealMLP	6.6999	6.8356	6.8266	6.7560	6.7897	6.8216
TabICLv2	6.7085	–	–	6.7128	6.7116	6.7050
TabM	6.6718	6.7279	6.7352	6.6726	6.7463	6.7925

diamonds, RMSE ($\times 10^{2}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	5.3510	5.2837	5.2290	5.3510	5.3640	5.2191
MLP†	5.4619	5.3203	5.2409	5.4619	5.2184	5.1342
RealMLP	5.3029	5.1619	5.1128	5.2234	5.1864	5.1271
TabICLv2	5.0300	–	–	5.0296	5.0147	5.0156
TabM	5.3306	5.2479	5.2501	5.3093	5.1712	5.1364

E-CommereShippingData, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.4093	7.4000	7.4178	7.4093	7.4021	7.4444
MLP†	7.4068	7.4200	7.4300	7.4068	7.4770	7.4240
RealMLP	7.4021	7.4581	7.4195	7.4398	7.4222	7.4256
TabICLv2	7.4254	–	–	7.4248	7.3968	7.4092
TabM	7.3870	7.3824	7.3788	7.3845	7.3845	7.4027

Fitness_Club, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.9310	8.0216	8.0378	7.9310	8.0316	7.9244
MLP†	8.0365	8.0919	8.0653	8.0365	8.0569	7.9745
RealMLP	8.0278	8.0191	8.0496	8.0622	8.0751	8.0547
TabICLv2	8.0934	–	–	8.0934	8.0921	8.0927
TabM	8.0586	8.0889	8.0879	8.0611	8.0511	8.0541

Food_Delivery_Time, RMSE ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.6611	7.4201	7.4147	7.6611	7.3634	7.3204
MLP†	7.8429	7.6235	7.6289	7.8429	7.5006	7.4710
RealMLP	7.9906	7.4363	7.4367	8.0071	7.4986	7.4641
TabICLv2	7.6692	–	–	7.6691	7.4975	7.4958
TabM	7.8406	7.7030	7.6909	7.8392	7.4391	7.4067

GiveMeSomeCredit, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.6923	8.7039	8.7058	8.6923	8.7029	8.7043
MLP†	8.7050	8.7042	8.7070	8.7050	8.7058	8.7112
RealMLP	8.6922	8.6970	8.6975	8.6912	8.6918	8.6994
TabICLv2	8.7077	–	–	8.7090	8.7091	8.6581
TabM	8.6999	8.7097	8.7085	8.6997	8.7003	8.7106

hazelnut-spread-contaminant-detection, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.6687	9.7066	9.7126	9.6687	9.7011	9.7131
MLP†	9.6966	9.7526	9.7618	9.6966	9.8457	9.8249
RealMLP	9.8212	9.8308	9.8249	9.7631	9.8176	9.8209
TabICLv2	9.9318	–	–	9.9319	9.9316	9.9342
TabM	9.5828	9.7999	9.7902	9.5883	9.8128	9.8204

healthcare_insurance_expenses, RMSE ($\times 10^{3}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	4.2355	4.1225	4.1011	4.2355	4.1674	4.0969
MLP†	4.2343	4.1473	4.1167	4.2343	4.0936	4.1049
RealMLP	4.1698	4.0538	4.0712	4.1121	4.0419	4.0563
TabICLv2	4.0425	–	–	4.0424	4.0289	4.0295
TabM	4.1350	4.1023	4.0973	4.1357	4.0737	4.0937

heloc, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.9342	7.9735	7.9804	7.9342	7.9784	7.9921
MLP†	7.9733	7.9626	7.9685	7.9733	7.9662	7.9915
RealMLP	7.9917	8.0150	8.0030	7.9903	7.9896	7.9952
TabICLv2	8.0094	–	–	8.0083	8.0009	8.0157
TabM	7.9655	7.9831	7.9862	7.9631	8.0019	8.0003

houses, RMSE ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	2.1883	2.1333	2.1322	2.1883	2.1534	2.1145
MLP†	2.2182	2.1999	2.1804	2.2182	2.1788	2.1286
RealMLP	2.2671	2.0184	2.0022	2.2291	2.0216	2.0065
TabICLv2	1.9616	–	–	1.9617	1.9267	1.9274
TabM	2.1477	2.0894	2.0635	2.1461	2.1090	2.0428

in_vehicle_coupon_recommendation, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.4203	8.5143	8.5228	8.4203	8.4663	8.4965
MLP†	8.3263	8.3917	8.4279	8.3263	8.4367	8.5114
RealMLP	8.4169	8.5675	8.5870	8.5102	8.5787	8.5886
TabICLv2	8.5062	–	–	8.5068	8.5214	8.5267
TabM	8.5377	8.5976	8.6178	8.5421	8.5650	8.5786

jm1, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.6036	7.6512	7.6593	7.6036	7.6130	7.6824
MLP†	7.3809	7.4401	7.4935	7.3809	7.3815	7.5575
RealMLP	7.4775	7.5538	7.7081	7.4593	7.4774	7.7164
TabICLv2	7.8442	–	–	7.7652	7.7652	7.8053
TabM	7.4297	7.3934	7.5481	7.4812	7.5949	7.6574

Marketing_Campaign, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.2370	9.3592	9.3713	9.2370	9.3636	9.3969
MLP†	9.2536	9.2760	9.2658	9.2536	9.2932	9.3512
RealMLP	9.3115	9.2394	9.3505	9.2485	9.2915	9.3500
TabICLv2	9.4026	–	–	9.4032	9.4032	9.4022
TabM	9.3547	9.3438	9.3343	9.3545	9.2946	9.3291

hiva_agnostic, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	1.7432	1.7444	1.7443	1.7414	1.7383	1.7377
MLP†	1.7526	1.7431	1.7414	1.7526	1.7700	1.7523
RealMLP	1.8713	1.7878	1.7672	1.7891	1.7428	1.7606
TabICLv2	1.9664	–	–	1.9670	1.7658	1.7528
TabM	1.7451	1.7392	1.7395	1.7456	1.7455	1.7440

HR_Analytics_Job_Change_of_Data_Scientists, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.1202	8.1392	8.1390	8.1202	8.1202	8.1354
MLP†	8.1177	8.1182	8.1207	8.1177	8.1222	8.1274
RealMLP	8.1188	8.1225	8.1302	8.1145	8.1178	8.1330
TabICLv2	8.1413	–	–	8.1216	8.1581	8.1544
TabM	8.1203	8.1092	8.1234	8.1185	8.1105	8.1349

Is-this-a-good-customer, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	6.9105	6.9239	7.2031	6.9105	6.9105	7.3272
MLP†	7.3582	7.2826	7.3951	7.3582	7.4105	7.2775
RealMLP	7.3418	7.4820	7.4445	7.5290	7.4052	7.4894
TabICLv2	7.4198	–	–	7.4172	7.4361	7.4103
TabM	7.4561	7.3793	7.4251	7.4504	7.1531	7.2308

kddcup09_appetency, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.9096	8.1183	8.1852	7.9096	8.1132	8.3270
MLP†	8.2343	8.2772	8.2675	8.2343	8.2643	8.3162
RealMLP	8.1917	8.2738	8.2545	8.2003	8.3143	8.2910
TabICLv2	8.1640	–	–	8.1640	8.1640	8.1711
TabM	8.0890	8.2447	8.2393	8.0912	8.2501	8.3064

maternal_health_risk, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	4.5202	4.2644	4.1954	4.5202	4.0941	4.1794
MLP†	5.8054	5.1471	5.2874	5.8054	4.0819	4.1936
RealMLP	5.9352	3.9439	4.0137	4.2008	3.8195	3.8588
TabICLv2	3.5423	–	–	3.5422	3.6822	3.5875
TabM	5.2205	4.3698	4.5208	5.1543	4.1690	4.0833

miami_housing, RMSE ($\times 10^4$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.8653	8.2412	8.1592	8.8653	8.4700	8.1959
MLP†	8.3538	8.2948	8.2816	8.3538	8.2661	8.1162
RealMLP	8.8388	8.2385	8.1793	8.9164	7.9579	7.8537
TabICLv2	7.9394	–	–	7.9377	7.8080	7.7871
TabM	8.4637	8.3457	8.1751	8.5195	8.0500	7.8909

MIC, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	5.2214	4.7241	4.7263	5.2242	4.6800	4.6989
MLP†	4.9399	4.6754	4.6779	4.9399	4.9304	4.6246
RealMLP	5.0755	4.5056	4.4439	4.7880	4.6474	4.5229
TabICLv2	4.6935	–	–	4.7008	4.6119	4.5926
TabM	4.4498	4.4495	4.4958	4.4451	4.7618	4.5774

NATICUSdroid, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.8439	9.8611	9.8605	9.8439	9.8595	9.8596
MLP†	9.8537	9.8561	9.8599	9.8537	9.8545	9.8654
RealMLP	9.8519	9.8627	9.8677	9.8557	9.8682	9.8692
TabICLv2	9.8721	–	–	9.8718	9.8756	9.8696
TabM	9.8597	9.8608	9.8639	9.8591	9.8635	9.8696

online_shoppers_intention, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.2878	9.2624	9.2842	9.2878	9.2749	9.2823
MLP†	9.2699	9.2489	9.2723	9.2699	9.2646	9.2935
RealMLP	9.2241	9.2668	9.2787	9.2562	9.3177	9.3122
TabICLv2	9.3270	–	–	9.3269	9.3262	9.3141
TabM	9.2932	9.2970	9.3048	9.2910	9.2869	9.3077

physiochemical_protein, RMSE ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	3.5064	3.4130	3.4165	3.5064	3.2734	3.2687
MLP†	3.4397	3.4329	3.3276	3.4397	3.4798	3.2655
RealMLP	3.4791	3.1899	3.1409	3.3598	3.2314	3.1467
TabICLv2	3.0272	–	–	3.0274	3.0274	3.0059
TabM	3.4450	3.3289	3.2605	3.4481	3.2896	3.2368

polish_companies_bankruptcy, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.6286	9.6375	9.6533	9.6286	9.6285	9.6589
MLP†	9.5066	9.5469	9.5899	9.5066	9.6337	9.6771
RealMLP	9.6986	9.6706	9.7341	9.6740	9.7277	9.8087
TabICLv2	9.8448	–	–	9.8435	9.8570	9.8419
TabM	9.6004	9.7586	9.7514	9.5977	9.6531	9.7139

qsar-biodeg, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.1701	9.1496	9.2257	9.1701	9.2617	9.2733
MLP†	9.2747	9.2927	9.2938	9.2747	9.3187	9.3263
RealMLP	9.3223	9.3079	9.3021	9.2935	9.2697	9.3072
TabICLv2	9.3551	–	–	9.3541	9.3411	9.3577
TabM	9.2740	9.2715	9.3285	9.2740	9.3025	9.3645

QSAR-TID-11, RMSE ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	8.3794	8.2195	8.1975	8.3794	8.1546	8.1625
MLP†	9.0223	8.4682	8.4104	9.0223	8.3292	8.1177
RealMLP	8.5030	8.2734	8.3142	8.3293	8.2608	8.2719
TabICLv2	8.7431	–	–	8.7467	8.6548	8.6310
TabM	8.3502	8.2649	8.2547	8.3527	8.1295	8.1447

QSAR_fish_toxicity, RMSE ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.2195	9.2261	9.1510	9.2195	9.2321	9.1103
MLP†	9.3012	9.2393	9.1841	9.3012	9.2756	9.4153
RealMLP	9.0203	9.2779	9.0148	9.0115	9.1212	8.9752
TabICLv2	8.9427	–	–	8.9424	8.9518	8.9638
TabM	9.3272	9.1361	9.0848	9.3213	9.2216	8.9853

SDSS17, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	0.8797	0.7217	0.7215	0.8797	0.7366	0.7309
MLP†	1.0092	0.8167	0.7931	1.0092	0.7888	0.7420
RealMLP	1.0577	0.8650	0.8582	1.0973	0.8542	0.8448
TabICLv2	0.7204	–	–	0.7199	0.6402	0.6329
TabM	0.9472	0.8181	0.8179	0.9492	0.7185	0.7107

seismic-bumps, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	7.7969	7.9525	7.9710	7.7947	7.9098	7.8211
MLP†	7.8723	7.9000	7.8838	7.8723	7.7816	7.9525
RealMLP	7.8653	7.8616	7.8169	7.9403	7.7415	7.8376
TabICLv2	7.8562	–	–	7.9974	7.9335	8.0798
TabM	7.8797	7.9693	7.9545	7.8788	7.7594	7.7701

splice, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	0.9156	0.8190	0.8290	0.9156	0.9156	0.9156
MLP†	1.2279	1.0903	1.1063	1.2279	0.9015	1.0049
RealMLP	0.9759	0.9163	0.8936	0.9242	0.9017	0.8628
TabICLv2	0.7394	–	–	0.7391	0.7355	0.7423
TabM	0.9813	0.9811	0.9468	0.9745	0.8700	0.8235

students_dropout_and_academic_success, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	5.6762	5.5890	5.5910	5.6762	5.6493	5.5995
MLP†	5.7487	5.7407	5.6818	5.7487	5.6952	5.6395
RealMLP	5.7781	5.6158	5.5592	5.7540	5.6478	5.5847
TabICLv2	5.5024	–	–	5.5023	5.5796	5.4955
TabM	5.6311	5.6390	5.6067	5.6344	5.5936	5.5316

superconductivity, RMSE ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.3300	9.1235	9.1389	9.3300	9.1403	9.1486
MLP†	9.2630	9.4117	9.2116	9.2630	9.3933	9.1269
RealMLP	9.4589	9.4891	9.1818	9.3690	9.1963	9.0773
TabICLv2	8.9416	–	–	8.9442	8.9420	8.9073
TabM	9.2824	9.1220	9.1136	9.2747	9.2128	9.1118

taiwanese_bankruptcy_prediction, AUROC ($\times 10^{-1}$) ↑

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	9.4585	9.4696	9.4689	9.4585	9.4636	9.4948
MLP†	9.4060	9.4070	9.3960	9.4060	9.4380	9.4598
RealMLP	9.4567	9.4963	9.5160	9.4635	9.4358	9.5194
TabICLv2	9.4634	–	–	9.4636	9.5219	9.5188
TabM	9.4132	9.3936	9.4310	9.3985	9.4700	9.4395

website_phishing, Log-loss ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	2.3792	2.3165	2.2906	2.3792	2.3918	2.3235
MLP†	2.4254	2.3392	2.3463	2.4254	2.2644	2.2324
RealMLP	2.2505	2.1099	2.1292	2.1869	2.1033	2.1082
TabICLv2	2.0356	–	–	2.0349	2.0143	2.0119
TabM	2.3206	2.2666	2.2364	2.3182	2.2823	2.2216

wine_quality, RMSE ($\times 10^{-1}$) ↓

	Base			Agentic		
Model	Def.	Tuned	T+E	Def.	Tuned	T+E
LightGBM	6.1768	5.9467	5.9440	6.1768	6.0843	5.9289
MLP†	6.4123	6.4301	6.1561	6.4123	5.9714	5.9587
RealMLP	6.1023	5.8850	5.8408	6.1381	5.8928	5.8470
TabICLv2	5.8776	–	–	5.8770	5.8514	5.8377
TabM	6.1193	6.0430	5.9683	6.1365	5.9932	5.8840