BEYOND FOUNDATION MODELS: PFN PRIORS IMPROVE TABULAR HYPERPARAMETER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic priors have so far served one purpose: pretraining tabular foundation models. We ask whether the same priors can improve conventional tabular models, which are trained from scratch on each dataset. We propose PriorTune, which meta-tunes hyperparameter configurations on synthetic tasks alone. Offline, for a given model, we fit a pool of configurations on datasets sampled from a published PFN prior and greedily select the subset with the highest winrate over the random baseline. Online, a new dataset fits only the selected configurations, so the gain comes at no extra cost. Empirically, we observe that scaling both the number of configurations and synthetic datasets has an equal scaling effect on the performance of the final subset of configurations, which we also explain theoretically. On BeyondArena, the models tuned with our hyperparameter configurations approach the performance of the recent TabPFN-3, while being much more efficient at inference. Overall, our results establish existing PFN priors as a scalable source of meta-training data for hyperparameter selection for non-foundational tabular models.

1 INTRODUCTION

Synthetic data drives the development of tabular foundation models. Since TabPFNv1 (Hollmann et al., 2023), every generation has come with a richer prior: TabPFNv2 added new edge types to the causal graph sampler (Hollmann et al., 2025), TabICLv2 added heavy-tailed random graphs and a filter on the generated datasets (Qu et al., 2026), and TabPFN-3 scaled the generated datasets to a million rows (Grinsztajn et al., 2026). Each step raised benchmark scores. To date, these priors have mostly served one purpose: pretraining PFNs.

Conventional models, which are trained from scratch on each dataset, have no direct way to use these priors. For such models, e.g. GBDTs and DNNs, hyperparameter selection is the natural way for transferring knowledge across tasks. TabRepo demonstrates that a set of hyperparameter configurations selected on approximately 200 real datasets transfers effectively to unseen datasets (Salinas & Erickson, 2024). Similarly, RealMLP identifies strong MLP architecture and training defaults through meta-tuning on 118 real datasets (Holzmüller et al., 2024). Both rely on a small, fixed collection of real datasets, which caps the scale and diversity of the meta-training tasks. This raises a natural question: can datasets sampled from a PFN prior replace real datasets when learning transferable sets of hyperparameter configurations? If so, conventional models can draw on synthetic priors without any change to how they are trained. We define a meta-objective for configuration set selection, develop an algorithm to optimize it, and scale the offline search to 4,000 candidate configurations evaluated across 4,000 synthetic datasets.

We study four efficient tabular models: LightGBM (Ke et al., 2017), LightGBM with TabPrep (Tschalzev et al., 2026), an MLP with cosine embeddings (Gorishniy et al., 2022), and TabPack (Gorishniy et al., 2026). These four models are near the efficiency-performance pareto frontier (Figure 1) and cover both tree-based and DNN-based approaches. We use TabICLv2 as our main synthetic prior, with ablations on TabPFNv1 and TabICLv1, and evaluate transfer on TabArena's 51 IID datasets (Erickson et al., 2025) and BeyondArena's 142 datasets with diverse splitting protocols (Purucker et al., 2026).

As a source of meta-training tasks, synthetic priors have three advantages over real datasets. First, supply is unlimited and quality is controllable. Real repositories plateau: on TabRepo's 182 datasets,

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.

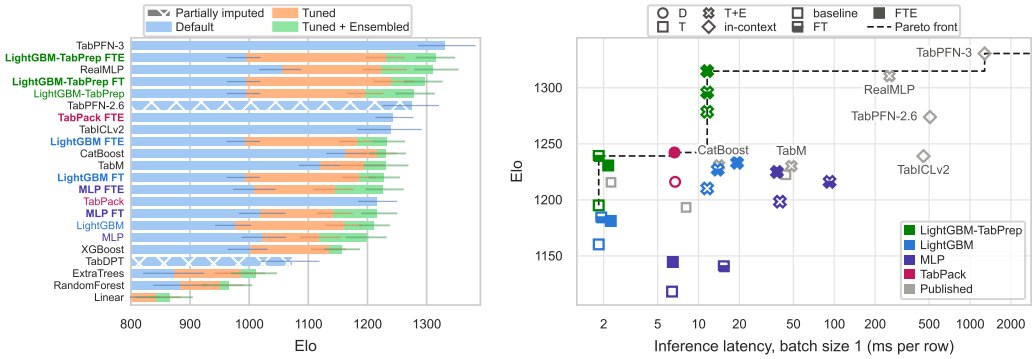


Figure 1: BeyondArena-Lite (142 datasets), one Elo field of 55 methods. **Left:** Elo under default, tuned, and tuned-and-ensembled deployment. Our families are coloured; FT and FTE are the subsets meta-tuned under T and T+E, and the uncoloured row of each family is its baseline (a random subset, or the published LightGBM). **Right:** Elo against inference latency at batch size 1, the median time of a single-row predict call on a fitted model kept in memory. GBDTs on 16 CPU threads, DNNs and PFNs on one A100; PFNs at default settings with internal ensembling (protocol in Appendix A.1).

adding configurations stops helping, whereas on synthetic tasks neither the number of tasks nor the number of configurations has saturated at 4,000 (Section 5). Second, the cost is paid once, offline, and we release the result. On a new dataset, the model trains only the K selected configurations, so the gain over random search adds nothing to training or inference cost. The offline bank also has lower compute cost than pretraining a single PFN (Appendix A.1). Third, synthetic tasks rule out test-set contamination by construction. Meta-training on real data requires an audit of overlap; for TabArena alone, we had to remove 18 of the 200 TabRepo datasets.

The contributions of this paper are as follows:

1. **Meta-tuning hyperparameters on synthetic data.** We show that existing PFN priors can improve conventional tabular models through offline hyperparameter selection, without training a PFN or changing the models' training procedures. Real datasets are worth more per task, but synthetic tasks can be scaled: at $4{,}000 \times 4{,}000$, subsets selected on synthetic tasks end above the best subsets selected on TabRepo for every model and every K.

2. **A deployment-aware selection objective.** We select configurations by their win rate against random subsets under the deployment rule, validation-best selection or greedy ensembling, scored on held-out test splits. The objective is scale-free across tasks and metrics, and selecting for ensembling yields a different subset than selecting for the single best member.

3. **Improved tuning efficiency on real datasets.** Across both benchmarks, the selected sets of configurations outperform random sets of the same size. They also outperform random sets of every size up to the full pool of 4,000, so offline selection improves both the quality and the efficiency of downstream tuning. We release the selected subsets for $K \in \{25, 64, 200\}$ as drop-in search spaces for LightGBM, LightGBM-TabPrep, and the MLP, together with the $4{,}000 \times 4{,}000$ score caches: 16 million fits per family, sixty times the number of evaluations in TabRepo.

4. **An analysis of scaling and prior choice.** We investigate how the quality of the configuration set depends on the number of synthetic tasks, the candidate pool size, and the generating prior, and derive a transfer bound and a record-count argument that motivate the observed scaling forms. Both tasks and configurations keep paying off at 4,000, whereas the choice of prior appears secondary: even the TabPFNv1 prior is competitive with TabICLv2.

2 Related Work

Transfer of hyperparameter configurations. Good hyperparameter configurations typically transfer across datasets. Feurer et al. (2015) warm-start Bayesian optimization with configurations that performed well on similar datasets. Later work drops the similarity and learns one fixed set for all datasets: given a table of scores for every configuration on every dataset, greedily add the configuration that most improves the average best-member score over datasets (Wistuba et al., 2015; Pfisterer et al., 2021; Winkelmolen et al., 2020). TabRepo scales this offline selection to 1,310 configurations on 200 datasets, and AutoGluon 1.0 ships the learned set as its default (Salinas & Erickson, 2024; Erickson et al., 2020). RealMLP meta-learns several model components (e.g. preprocessing, activations) on 118 datasets (Holzmüller et al., 2024). All of this work is bounded by the supply of real data, 25 to 208 datasets. TabRepo reports a saturating effect around 150 datasets or configurations. We keep the greedy forward selection, but run it on synthetic tasks, whose supply is unlimited, and change the scoring function (Section 3.4).

A second line transfers knowledge into the optimizer rather than into a set of configurations. One line of work fits the Gaussian-process prior on evaluations from previous tasks (Perrone et al., 2018; Feurer et al., 2018; Wistuba & Grabocka, 2021; Wang et al., 2024; Fan et al., 2023). OptFormer pretrains a text transformer on optimization logs (Chen et al., 2022). Recent works such as PFNs4BO and its successors, PFNs pretrained on synthetic optimization tasks (Müller et al., 2023; Yu et al., 2026; Rogers & Ponnada, 2026; Zhang et al., 2026). All of these search sequentially and need the surrogate at tuning time. Our configuration set is built once offline and can be evaluated in parallel.

Synthetic priors for tabular data. Synthetic priors are the engine of tabular foundation models. Among others, the recent TabPFN, TabICL, Mitra, LimiX, Xiaomi-TabLDM, Causilo, and EXAONE Tabular are pretrained on synthetic tables alone (Hollmann et al., 2023; 2025; Grinsztajn et al., 2026; Qu et al., 2025; 2026; Zhang et al., 2025b;a; Wang et al., 2026; Cho et al., 2026; Eo et al., 2026). A typical synthetic prior has three parts: a directed graph over variables, an edge function that maps parents to a child, and an observation process. Graphs are random DAGs in TabPFN, Cauchy random graphs with varied connectivity in TabICLv2, and hierarchical causal models in LimiX. Edge functions started as small neural networks and grew to include decision trees and discretization in TabPFNv2, tree ensembles in TabICL, and Gaussian processes, polynomials, and products in TabICLv2 (Hollmann et al., 2025; Eo et al., 2026). Related in spirit, recent work distills a pretrained tabular foundation model into a tree ensemble or an MLP, one dataset at a time (Tanna et al., 2026; Qin et al., 2026); this needs the foundation model at training time on every dataset. We use the prior once, offline, and the deployed model is unchanged.

3 Method

In this section, we present PriorTune, a method for meta-tuning strong hyperparameter configurations of conventional tabular models on synthetic data (Figure 2). Offline, we sample N tasks from a synthetic prior $\mathcal{Q}$ (Section 3.2), fit a pool $\mathcal{C}$ of C configurations once per task (Section 3.3), and greedily select K of them (Section 3.4). Online, a real task fits only the K selected configurations and deploys them by validation, either the best one (*tuned*) or a greedy ensemble (*tuned and ensembled*).

3.1 Setup

Notation. We consider regression and classification on tabular data. We denote the synthetic data generator by $\mathcal{Q}$ and a dataset by D. The loss ℓ is the native metric of the task, oriented so that lower is better: RMSE for regression, log-loss for multiclass, and negated ROC-AUC for binary classification. We split each dataset into train, validation, and test subsets $D_{\text{train}}, D_{\text{val}}, D_{\text{test}}$.

Benchmarks. We use TabArena (Erickson et al., 2025) and BeyondArena (Purucker et al., 2026), which together cover 155 to 1.3 million rows, 3 to 22,215 features, and i.i.d., grouped, and temporal splits. Table 1 summarizes both suites. For both, we take the Lite version: the first train-test split and up to 8 inner train-validation splits. No benchmark data enters the selection.

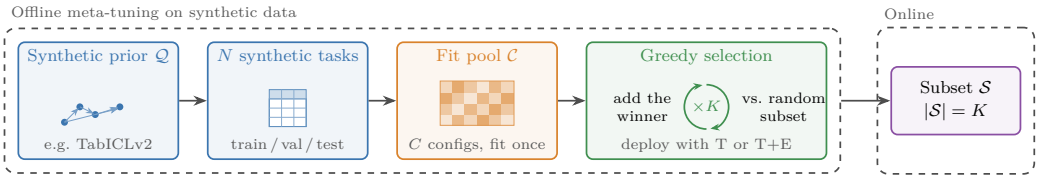


Figure 2: Overview. Offline, configurations from a pool $\mathcal{C}$ are fitted once on N synthetic tasks and a subset of K is selected greedily by win rate over random subsets under the deployment rule. Online, a real task fits only the K selected configurations.

Table 1: Benchmark statistics. Row and feature counts are per dataset (minimum / median / maximum).

	TabArena	BeyondArena
Datasets	51	142
Binary / multiclass / regression	30 / 8 / 13	73 / 25 / 44
Splits: i.i.d. / grouped / temporal	51 / 0 / 0	103 / 18 / 21
Rows	748 / 6,497 / 150,000	155 / 5,000 / 1,296,316
Features	4 / 17 / 1,776	3 / 23 / 22,215

3.2 Synthetic data generation

For all main experiments, we sample tasks from the TabICLv2 prior (Qu et al., 2026). Three settings of the prior matter most, and we keep them close to the third pretraining stage of TabICLv2. Dataset size is log-uniform between 400 and 60,000 rows. The feature count is uniform between 1 and a cap that falls from 100 to 20 as the row count grows. The sampled datasets are filtered using both of the available filters: reject if the target node of the graph shares no ancestors with feature nodes, and reject if ExtraTrees cannot beat the constant predictor. We fix the task mix at 2,000 regression, 1,400 binary, and 600 multiclass datasets with 3 to 10 classes, and split each dataset 64/16/20 into train, validation, and test, stratified for classification. Appendix A.3 lists all settings.

3.3 Models

For our main runs, we use LightGBM (Ke et al., 2017), LightGBM with TabPrep (Tschalzev et al., 2026), MLP with cosine embeddings (Gorishniy et al., 2022), and TabPack (Gorishniy et al., 2026). We select these models for their tradeoff between training and inference speed and performance (Figure 1). Each configuration in the pool $\mathcal{C}$ with $C = 4{,}000$ is an independent draw from the model's published search space, which we list in Appendix A.2. TabPack is a packed ensemble of MLPs, so its unit of sampling is a single MLP configuration: it shares the pool of $C = 4{,}000$ MLP configurations, from which we select its $K = 64$ members.

3.4 Subset selection

The selection mimics the deployment rule and scores against random subsets of the same size. Following TabArena and BeyondArena, we distinguish two deployment rules: *tuned* (T), where the validation-best member is deployed, and *tuned and ensembled* (T+E), where validation builds a greedy ensemble (Caruana et al., 2004). We implement the whole procedure as a greedy forward selection algorithm: at each iteration starting from an empty set, we add a configuration, score it under the given deployment rule, and select the one that yields the highest winrate against the random subset (Algorithm **??**). Appendix B states the objective formally. Below we motivate each choice on a small example.

Table 2 gives a toy instance under the tuned rule: four configurations A to D, three synthetic tasks D_1 to D_3 for selection, one held-out task D_4 standing for a real dataset, and a subset size of $K = 2$.

Table 2: A toy example under the *tuned* rule. Four configurations, three synthetic datasets D_1–D_3 used for selection (D_1 scored by RMSE, the rest by log-loss) and one held-out dataset D_4 (log-loss). Cells are validation / test losses (lower is better); the last row is the test loss of the deployed member of a random reference subset. We select a subset of $K = 2$ from the empty set.

	D_1		D_2		D_3		D_4, held out	
Configuration	val	test	val	test	val	test	val	test
A	105	103	0.28	0.36	0.60	0.45	0.38	0.47
B	96	96	0.30	0.31	0.53	0.53	0.41	0.41
C	95	97	0.34	0.34	0.55	0.55	0.40	0.40
D	60	60	0.29	0.40	0.52	0.60	0.39	0.50
Random reference	–	100	–	0.32	–	0.50	–	0.42

Algorithm 1 PriorTune subset selection

Require: cached validation and test predictions of every $c \in \mathcal{C}$ on synthetic tasks $D_1, \ldots, D_N$; subset size K; number of draws R; deployment rule $\mathcal{E} \in \{\mathrm{T}, \mathrm{T{+}E}\}$ equation 1

Draw references $\mathcal{R}_1, \ldots, \mathcal{R}_R \sim \mathrm{Unif}(\Pi_K)$ and, for every task i, bootstrap replicas $D^{(1)}_{\mathrm{val},i}, \ldots, D^{(R)}_{\mathrm{val},i}$ of its validation rows
$\mathcal{S}_0 \leftarrow \emptyset$
for $k = 1, \ldots, K$ **do**
  for each candidate $c \in \mathcal{C} \setminus \mathcal{S}_{k-1}$ **do**
    # deploy both with $\mathcal{E}$ on replica r, score on test; $h(a,b) = \mathbf{1}_{a<b} + \frac{1}{2}\mathbf{1}_{a=b}$
    $\widetilde{U}(c) \leftarrow \dfrac{1}{NR} \displaystyle\sum_{i=1}^{N} \sum_{r=1}^{R} h\big(L_i^{(r)}(\mathcal{S}_{k-1} \cup \{c\}),\, L_i^{(r)}(\mathcal{R}_r)\big)$
  end for
  $c_k \leftarrow \arg\max_c \widetilde{U}(c)$, ties broken by the lowest configuration id
  $\mathcal{S}_k \leftarrow \mathcal{S}_{k-1} \cup \{c_k\}$
end for
return $\mathcal{S}_K$

Selecting on validation, scoring on test. Under both deployment rules, the model (or ensemble) is selected on the validation split and scored on the test split. Our objective mimics this, which selects configurations that are stable across validation and test. The alternative, selecting on test directly or on a single out-of-fold score as TabRepo and AutoGluon do (Salinas & Erickson, 2024), hides that instability. In Table 2, A and D are unstable: their validation and test losses disagree on most datasets, whereas B and C are stable. Both selections pick B first, then diverge: test-only selection returns $\{\mathrm{B}, \mathrm{A}\}$, validation-then-test returns the stable $\{\mathrm{B}, \mathrm{C}\}$. On the held-out D_4, $\{\mathrm{B}, \mathrm{A}\}$ deploys A and loses; $\{\mathrm{B}, \mathrm{C}\}$ deploys C and wins.

Comparing against the random baseline. The alternative is to use losses directly: TabRepo adds the configuration that most lowers the mean error of the best member over datasets (Salinas & Erickson, 2024). Raw losses mix metrics and scales across tasks. In Table 2, any rule on raw losses selects D first, since it cuts the RMSE on D_1 by 36 points, which outweighs every other gain in the table, although D has the worst test loss on every other dataset. Ranking objectives are preferred over regression objectives for the same reason in offline model-based optimization (Tan et al., 2025).

Stability. The third design choice is to resample both the validation split and the random reference. A single validation split decides which member is deployed: in Table 2, C beats B on the validation split of D_1 by one RMSE point, and another validation sample may deploy B instead. We therefore draw $R = 16$ bootstrap resamples of the validation rows, paired with R independent random references $\mathcal{R}_r$; with cached predictions the cost is negligible, and averaging over references is unbiased with variance $O(1/R)$ (Proposition 4).

Table 3: Median improvement over the tuned MLP with a random subset on BeyondArena-Lite (142 datasets), in percent of its error, by deployment rule and dataset size. FT and FTE are the subsets meta-tuned under T and T+E; Random is a subset of 25 random draws from the same search space (64 for TabPack).

		Median improvement over tuned MLP (%)							
		Tuned (T)				Tuned + ensembled (T+E)			
Model	Subset	Small	Medium	Large	All	Small	Medium	Large	All
MLP	Random	0.00	0.00	0.00	0.00	0.00	+1.42	+0.76	+0.67
	FT	+1.35	+0.17	−0.03	+0.16	+1.32	+1.15	+0.63	+0.93
	FTE	+1.05	+0.09	−0.01	+0.16	+1.53	+1.28	+0.83	+1.02
LightGBM	Random	+0.26	+0.37	+0.44	+0.41	+0.11	+0.97	+0.84	+0.75
	FT	0.00	+1.02	+0.66	+0.69	−0.29	+1.25	+0.80	+0.99
	FTE	−0.45	+1.10	+0.49	+0.50	+0.85	+1.28	+0.80	+0.82
LightGBM-TabPrep	Random	−0.52	+1.51	+0.74	+0.72	+0.73	+2.19	+1.56	+1.57
	FT	+0.24	+2.30	+0.78	+0.96	0.00	+2.33	+1.32	+1.67
	FTE	−0.45	+2.00	+1.20	+0.96	+0.20	+2.35	+1.76	+1.75
TabPack	Random	–	–	–	–	+0.87	+1.69	+0.85	+1.05
	FTE	–	–	–	–	+1.19	+1.53	+1.14	+1.25
TabPFN-3	Default	–	–	–	–	+7.69	+5.22	−0.31	+3.62
RealMLP	Random	+2.15	+1.11	+0.61	+1.08	+2.80	+2.22	+1.26	+1.69

Buckets by number of rows: small <2.5k (61 datasets), medium 2.5k–50k (42), large ≥50k (39). The reference is MLP Random under T. The LightGBM Random row is the benchmark's published run; the other Random rows are our own runs. TabPack and TabPFN-3 have a single deployment (TabPack's online ensemble over its members; TabPFN-3's default) and are listed under T+E.

4 Main results

Setup. We evaluate on BeyondArena-Lite (142 datasets) under the benchmark's protocol: 8-fold bagging on the first outer split, native metrics, and $K = 25$ configurations per dataset (the official budget); TabPack fits its $K = 64$ members. FT denotes the subset meta-tuned for T and FTE the subset meta-tuned for T+E; we report both under both rules. Random is a subset of 25 draws from the same search space: the benchmark's published run for LightGBM, our own runs for the MLP and LightGBM-TabPrep. Figure 1 shows Elo. Table 3 reports the per-dataset improvement over the tuned MLP with a random subset, $100\,(\ell_{\text{base}} - \ell)/\ell_{\text{base}}$, as a median over all datasets and within three size buckets. Unlike Elo, this metric does not depend on the pool of compared models. TabPFN-3 (Grinsztajn et al., 2026) and RealMLP (Holzmüller et al., 2024) are references taken from the benchmark's published runs.

Meta-tuned subsets beat random ones in every family and under both rules. Over all 142 datasets, the meta-tuned subset is ahead of the random subset of its own family in all seven family-rule pairs: by 0.3 points under T and 0.4 under T+E in the median improvement, with paired win rates of 0.55 to 0.68 (Table 8). Some gains are modest, but the direction is consistent across families, rules, and benchmarks, and it costs nothing at deployment. Section 5.2 shows that random subsets of any size up to the full pool of $4{,}000$ do not close the gap.

Selecting for the deployment rule. Matching the selection rule to the deployment rule helps or ties in five of six cases. Under T+E, the subset selected for T+E is ahead of the subset selected for T by 0.1 points for the MLP and LightGBM-TabPrep, and behind for LightGBM. Under T, the subset selected for T is ahead for LightGBM and ties for the MLP and LightGBM-TabPrep. The effect is small next to the gain from meta-tuning itself, but it is cheap: switching the deployment rule, K, or objective reuses the bank and costs a few hours on one GPU (Appendix A.1). Every result below shares the same $4{,}000 \times 4{,}000$ fits.

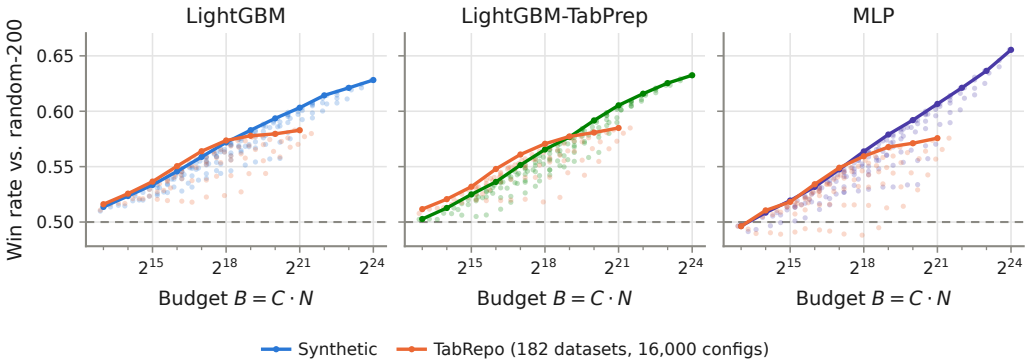


Figure 3: TabArena, T deployment, win rate against a random subset of the same size. Synthetic versus TabRepo meta-training data at $K = 200$. Dots are (C, N) cells; lines are the best cell within each factor-of-two budget bin. For TabRepo (182 datasets), we evaluated 16,000 configurations, so its curve stops at $B = 16{,}000 \times 182$.

Against the references. Under T+E over all datasets, LightGBM-TabPrep with the FTE subset reaches $+1.75$, against $+1.69$ for RealMLP and $+3.62$ for TabPFN-3. In the paired comparison, LightGBM-TabPrep ties with RealMLP at 52% wins and loses to TabPFN-3 on 59% of the datasets. RealMLP's defaults and search space were meta-tuned on 118 real datasets (Holzmüller et al., 2024), so matching it with synthetic tasks alone corroborates the usefuless of the synthetic data. TabPFN-3 owes its lead to the small and medium datasets, where in-context learning is strongest. On the 39 large datasets, LightGBM-TabPrep with the FTE subset under T+E is the strongest entry in Table 3: it gains $+1.76$ and beats TabPFN-3 on 69% of the datasets at a fraction of its inference cost, 12 ms per row against 1.3 s (Figure 1).

5 Ablations

All ablations use subsets selected under T and report the win rate of the meta-tuned subset against random subsets of the same size on TabArena (details in Appendix A.1). We use T because T+E selection runs a Caruana ensemble per candidate at every step, which is expensive. TabPack is absent: it is an online ensemble of MLP members, so the MLP pool covers it. Error bars are ± 1.96 standard errors over 2,000 selection repetitions on the fixed 51-dataset benchmark, not uncertainty over new datasets (Appendix A.1). The main text shows $K = 200$, TabArena's default tuning budget, and we include $K = 25, 64$ in Appendix A.4.

5.1 Synthetic versus real meta-training data

TabRepo (Salinas & Erickson, 2024) is the standard real-data source for meta-tuning: 200 datasets, of which we drop 18 that overlap with or share a source with TabArena (listed in Appendix A.4). We fit 16,000 configurations (our pool of 4,000 and 12,000 further draws) on the 182 datasets and select on them with the same protocol. Figure 3 compares the two sources at equal budget $B = C \cdot N$. Per task, real data is worth more at smaller budgets: up to 2^{18} subsets selected on the synthetic data show lower win rates, but from 2^{19} to 2^{20} on, synthetic data overtakes TabRepo. TabRepo saturates: over its last two doublings, to 16,000 configurations, its curve gains at most 0.008. We attribute this to the TabRepo size: with 182 datasets, the subsets start to overfit the source data pool. Synthetic data allows us to sidestep this, because we can continue scaling the data size indefinitely: at 2^{24} it ends 0.02 to 0.08 above TabRepo's best for every model and every K. At $K = 25$ and 64 the crossover lies between 2^{20} and 2^{22} (Appendix A.4).

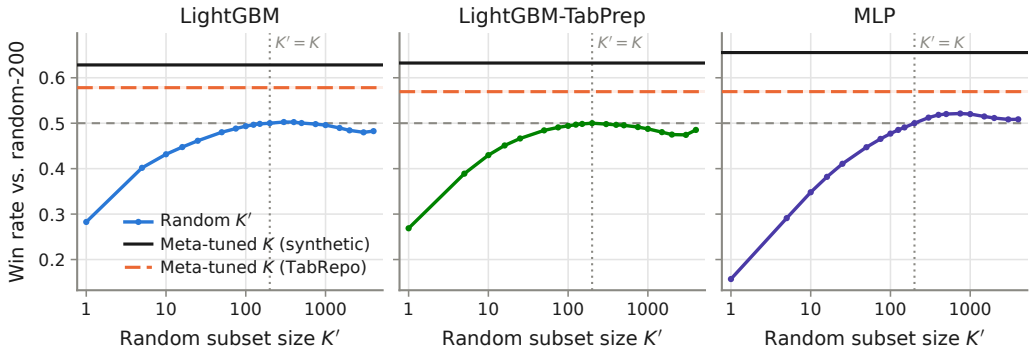


Figure 4: TabArena, T deployment, $K = 200$. Random subsets of growing size K' against the random-K baseline. Horizontal lines are the K configurations meta-tuned on the full $4{,}000 \times 4{,}000$ synthetic pool and on $4{,}000 \times 182$ TabRepo, against the same baseline.

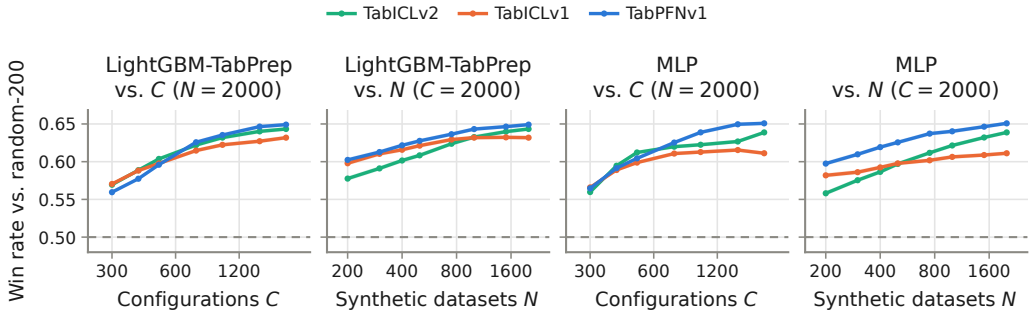


Figure 5: TabArena, T deployment, win rate against a random subset of the same size, $K = 200$. Three priors on $2{,}000 \times 2{,}000$ pools for LightGBM-TabPrep and the MLP, as C or N grows with the other fixed.

5.2 Meta-tuning versus more random search

A cheaper alternative to meta-tuning is to fit more random configurations. Figure 4 scores random subsets of size K' against the random-K baseline. The curve rises, saturates at 0.50 to 0.52 for K' between 200 and 750, and falls to 0.48 to 0.51 at the full pool: larger pools overfit the validation split. The meta-tuned subset of 200 sits at 0.63 to 0.66, above every random subset up to the full pool; the subset selected on TabRepo sits at 0.57. At $K = 25$ and 64, the random-K' curve peaks higher, at 0.51 to 0.60, and the meta-tuned subset still sits at least 0.08 above it (Appendix A.4). For a practitioner with a fixed budget, the 200 meta-tuned configurations replace a random search over the full pool that fits twenty times more models and still scores lower.

5.3 Choice of prior

We repeat the selection on synthetic tasks from the TabICLv1 (Qu et al., 2025) and TabPFNv1 (Hollmann et al., 2023) priors (Appendix A.3), with 2,000 configurations on 2,000 datasets for LightGBM-TabPrep and the MLP (Figure 5). At the full pool, the three priors are within 0.02 of each other for LightGBM-TabPrep at every K. For the MLP, TabICLv2 wins at $K = 25$ and 64 and TabPFNv1 at $K = 200$. Even TabPFNv1, the oldest and simplest prior, is competitive. The spread between priors, at most 0.04, equals one to four doublings of the number of synthetic tasks, so the generator is secondary to the number of tasks. We use TabICLv2, the newest prior, for the main experiments.

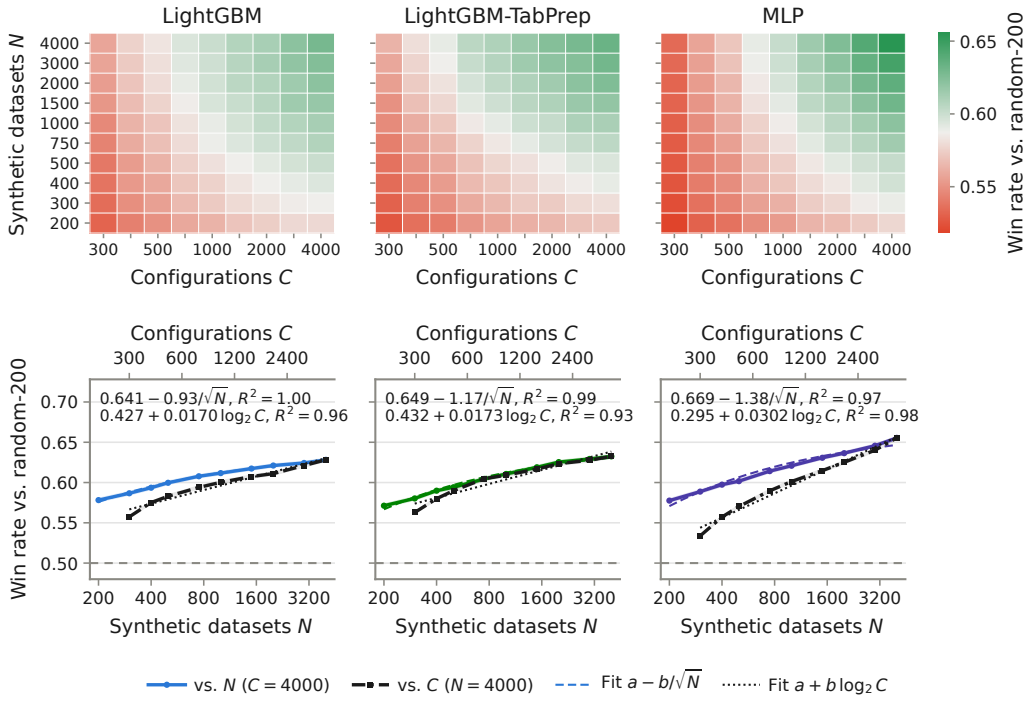


Figure 6: TabArena, T deployment, win rate against a random subset of the same size, $K = 200$. Top: the (C, N) grid. Bottom: against N at $C = 4{,}000$ (bottom axis) and against C at $N = 4{,}000$ (top axis), both on the same doubling scale, with the fits $a - b/\sqrt{N}$ and $a + b \log_2 C$.

5.4 Scaling

Figure 6 shows the win rate over the full (C, N) grid. It rises at almost every step along both axes, and neither axis has saturated at $4{,}000$. Appendix B motivates two forms: the estimation error over N synthetic tasks decays as $1/\sqrt{N}$ (Proposition 2), and the number of improvements available in a pool grows as $\log C$ (Proposition 1). We fit both forms to the full-pool curves and obtain R^2 between 0.92 and 1.00 for $a - b/\sqrt{N}$ and between 0.90 and 0.98 for $a + b \log_2 C$, over all three models and all K (Table 9). On the shared doubling scale of Figure 6, the curve in C is steeper than the curve in N for every model: at $K = 200$, a doubling of C is worth 1.2 to 1.7 doublings of N, and the curve in N tends to flatten above about 1,000 tasks. Once there are a few thousand synthetic tasks, more configurations are the better use of a fixed budget. Tasks reduce the estimation error of the objective, whereas configurations add candidates worth finding.

6 Conclusion

We have shown that the synthetic priors built for PFN pretraining are a scalable source of meta-training tasks for conventional tabular models. PriorTune selects a subset of hyperparameter configurations on synthetic tasks alone, by win rate against random subsets under the deployment rule. The selected subsets beat random subsets of any size, improve LightGBM, LightGBM-TabPrep, the MLP, and TabPack on the TabArena and BeyondArena benchmarks, and set a new state-of-the-art among non-foundation models on BeyondArena by Elo and median improvement, at lower latency than the previous best. The released subsets are drop-in search spaces: a practitioner replaces random search over the published space with the K selected configurations.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code/Codex) to implement parts of the experiment and analysis code. We have not used generative AI tools to generate synthetic datasets (the synthetic tasks come from the published priors of Section 3.2), to develop the theoretical framework, to formulate or prove mathematical claims, or to propose hypotheses or design the methodology and experiments. Additionally, we used generative AI tools to draft parts of the paper, to edit the text for readability and concision, and to search for and format references. We have reviewed all AI-assisted work: the code was checked and tested by the authors, every number in the paper is computed by scripts from the logged experiment results and was cross-checked against those logs, all references were verified against their sources, and all text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 specify the selection procedure and its fixed settings ($R = 16$, tie rule, 40 Caruana steps). Appendix A.1 gives the evaluation protocol on both benchmarks, the repetition scheme of the ablations, the compute used, and the latency measurement; Appendix A.2 lists the complete hyperparameter search space of every model; Appendix A.3 describes the three synthetic priors and Table 7 the exact sampling settings and seeds; Appendix A.4 lists the TabRepo datasets excluded for overlap with TabArena. Appendix B states all assumptions and gives complete proofs of the propositions. The benchmarks (TabArena, BeyondArena, TabRepo) are public. The supplementary material contains the code for synthetic task generation, fitting, selection, and evaluation, the selected subsets for every model and K, the seeds, and the score caches of all fits, from which every table and figure in the paper can be regenerated with the included scripts.

REFERENCES

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, 2004.

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 (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.13320.

Minyong Cho, Minho Jeong, Dooho Lee, Jinmo Lee, and Jaemin Yoo. Causilo technical report. *arXiv preprint arXiv:2609.22866*, 2026. URL https://arxiv.org/abs/2609.22866.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE Tabular 1.0: Technical report. *arXiv preprint arXiv:2608.25774*, 2026. URL https://arxiv.org/abs/2608.25774.

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

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025. URL https://arxiv.org/abs/2506.16791.

Zhou Fan, Xinran Han, and Zi Wang. HyperBO+: Pre-training a universal prior for Bayesian optimization with hierarchical Gaussian processes. *arXiv preprint arXiv:2212.10538*, 2023. URL https://arxiv.org/abs/2212.10538. Full version of a NeurIPS 2022 workshop paper.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1128–1135, 2015. doi: 10.1609/aaai.v29i1.9354.

Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219*, 2018. URL https://arxiv.org/abs/1802.02219.

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, Akim Kotelnikov, Ivan Rubachev, and Artem Babenko. TabPack: Efficient hyperparameter ensembles for tabular deep learning. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2607.05380.

Léo Grinsztajn, Klemens Flöge, Oscar Key, et al. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

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)*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637:319–326, 2025.

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 (NeurIPS)*, 2024.

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

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25444–25470, 2023. URL https://proceedings.mlr.press/v202/muller23a.html.

Valerio Perrone, Rodolphe Jenatton, Matthias W. Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Florian Pfisterer, Jan N. van Rijn, Philipp Probst, Andreas C. Müller, and Bernd Bischl. Learning multiple defaults for machine learning algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 241–242, 2021. doi: 10.1145/3449726.3459523. Extended version: arXiv:1811.09409.

Lennart Purucker, Andrej Tschalzev, Nick Erickson, Blayer, David Holzmüller, Arazi, Pfefferle, Tajjar, Gaël Varoquaux, and Frank Hutter. Beyond IID: How general are tabular foundation models, really? *arXiv preprint arXiv:2606.30410*, 2026.

Qi Qin, Jiajie Zhu, Dali Chen, Yuzhao Zhang, Jia-Xing Han, Peng Zhang, Ying Yan, Yifan Sun, and Yu Su. GEAR: Generative expansion and real anchoring for two-stage distillation of tabular foundation models. *arXiv preprint arXiv:2608.18849*, 2026. URL https://arxiv.org/abs/2608.18849.

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. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2602.11139.

Alfréd Rényi. Théorie des éléments saillants d'une suite d'observations. *Annales scientifiques de l'Université de Clermont-Ferrand 2*, 8(2):7–13, 1962.

Theodore Rogers and Srividya Ponnada. Zero-shot Bayesian optimization with TabPFN: Competitive with state-of-the-art without per-task training. In *International Conference on Automated Machine Learning (AutoML)*, 2026. Listed on the Amazon Science publications page; no arXiv version found.

David Salinas and Nick Erickson. TabRepo: A large scale repository of tabular model evaluations and its AutoML applications. In *Proceedings of the Third International Conference on Automated Machine Learning*, volume 256 of *Proceedings of Machine Learning Research*, pp. 19/1–30, 2024. URL https://proceedings.mlr.press/v256/salinas24a.html.

Rong-Xi Tan, Ke Xue, Shen-Huan Lyu, Haopu Shang, Yao Wang, Yaoyuan Wang, Sheng Fu, and Chao Qian. Offline model-based optimization by learning to rank. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.11502.

Aditya Tanna, Nassim Bouarour, Mohamed Bouadi, Vinay Kumar Sankarapu, and Pratinav Seth. Pocket foundation models: Distilling TFMs into CPU-ready gradient-boosted trees. *arXiv preprint arXiv:2605.18654*, 2026. URL https://arxiv.org/abs/2605.18654.

Andrej Tschalzev, Nick Erickson, Yuyang Wang, Huzefa Rangwala, Stefan Lüdtke, Heiner Stuckenschmidt, and Christian Bartelt. TabPrep: Closing the feature engineering gap in tabular benchmarks. *arXiv preprint arXiv:2606.02384*, 2026.

Penghui Wang, Wei Liu, Hong Wang, Chengyue Huang, Yuxi Sun, Zirui Wang, Hongming Huang, Quan Wang, Zhenwei Xin, Ping Hou, Jie Yu, Chunxiao Liu, Erli Meng, and Bin Wang. XiaomiTabLDM: A tabular foundation model technical report. *arXiv preprint arXiv:2609.03880*, 2026. URL https://arxiv.org/abs/2609.03880.

Zi Wang, George E. Dahl, Kevin Swersky, Chansoo Lee, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-trained Gaussian processes for Bayesian optimization. *Journal of Machine Learning Research*, 25(212):1–83, 2024. URL https://arxiv.org/abs/2109.08215.

Fela Winkelmolen, Nikita Ivkin, H. Furkan Bozkurt, and Zohar Karnin. Practical and sample efficient zero-shot HPO. *arXiv preprint arXiv:2007.13382*, 2020. URL https://arxiv.org/abs/2007.13382.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2101.07667.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Sequential model-free hyperparameter tuning. In *IEEE International Conference on Data Mining (ICDM)*, pp. 1033–1038, 2015. doi: 10.1109/ICDM.2015.20.

Rosen Ting-Ying Yu, Cyril Picard, and Faez Ahmed. GIT-BO: High-dimensional Bayesian optimization with tabular foundation models. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2505.20685.

Xingxuan Zhang, Gang Ren, Han Yu, et al. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025a. URL https://arxiv.org/abs/2509.03505.

Xinyu Zhang, Conor Hassan, Julien Martinelli, Daolang Huang, and Samuel Kaski. In-context multi-objective optimization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2512.11114.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b. URL https://arxiv.org/abs/2510.21204.

APPENDIX CONTENTS

A Computation **14**

A.1 Setup 14

A.2 Hyperparameter search spaces 14

A.3 Synthetic priors 16

A.3.1 Graph structure 16

A.3.2 Edge functions 16

A.3.3 Transformations and filters 17

A.3.4 Sampling settings 17

A.4 Additional results 17

B Theory **24**

B.1 Setup and main propositions 24

B.1.1 Setup 24

B.1.2 Upper bound on the improvement 26

B.1.3 Transfer from synthetic to real tasks 26

B.1.4 Analysis of the forward-selection heuristic 27

B.2 Proofs 27

A COMPUTATION

A.1 SETUP

Main results. On BeyondArena-Lite, every method is bagged over the 8 inner folds of the first outer split, and each configuration is trained once per fold. T selects the validation-best configuration, T+E fits a 40-step Caruana ensemble on the same validation predictions; both score the held-out test split with the native metric. FT subsets are selected with the T objective and FTE subsets with the T+E objective (40 Caruana steps inside the selection), both with $R = 16$ on the full $4{,}000 \times 4{,}000$ synthetic pool; no benchmark data enters the selection. TabPack's FTE subset is the $K = 64$ subset selected with the T+E objective on the MLP pool and deployed as one pack of those members.

Compute. We fit MLPs using tensorized code from TabPack (Gorishniy et al., 2026). Hence, one MLP fit takes on average 0.1 A100-seconds on the synthetic datasets, so the 16 million fits of the $4{,}000 \times 4{,}000$ bank cost 400 A100-hours. The real-data banks add around 100 A100-hours for TabRepo and TabArena. One LightGBM-TabPrep fit uses approximately 110 core-seconds, so its bank cost about 500,000 core-hours. The real data banks (TabRepo and TabArena) cost 100,000 core-hours. The two $2{,}000 \times 2{,}000$ banks of the prior ablation cost a quarter of this per family. For reference, pretraining one TabICLv2 model takes 24.5 H100 GPU-days, about 1,200 A100-hours (Qu et al., 2026), TabICL 1,440 A100-hours (Qu et al., 2025), and TabPFNv2 336 hours on 8x RTX 2080 Ti (Hollmann et al., 2025).

Inference latency. Latency (Figure 1) comes from the BeyondArena benchmark datasets. For each method and dataset, the model is fitted once and kept in memory. A single-row predict call on a random test row is then timed after 5 warm-up calls, for 50 calls. The timed call includes the model's own preprocessing and, on GPU, a device synchronization between CPU and GPU. GBDTs run on 16 CPU threads; DNNs and PFNs run on one NVIDIA A100 (80 GB) GPU. For PFN models, we use default settings which include the internal ensembling. Under T we time a single best configuration, under T+E we time all the members of the ensemble. TabPack tensorizes the prediction loop over its members on GPU, so it outperforms the naive sequential or parallel loop over MLP members. In-context models re-encode the training context on every call: TabPFN-3 takes 0.7 s per row at 4,000 training rows and 43 s at 93,000.

Ablations. Each ablation cell (C, N) runs $2{,}000$ repetitions. A repetition permutes the pool and the synthetic tasks, takes the first C configurations and N tasks, selects K of them by Algorithm 1 with $R = 16$, and scores the selected subset on TabArena under T against the first K configurations of the same permutation: a uniform random K-subset, which also serves as the first reference $\mathcal{R}_1$ inside the selection, so that the $C = K$ cell reads 0.5 by construction. $\mathcal{R}_1$ is still a uniform draw, so Algorithm 1 and Proposition 4 are unchanged. Validation resamples are drawn once per synthetic task and shared across repetitions. Ties in validation selection and in the greedy step go to the lowest configuration id. Every curve and heatmap in Figures 3–6 and in Appendix A.4 is the mean over these repetitions, with ± 1.96 standard errors; they measure variation across repetitions on the fixed benchmark, not uncertainty over new datasets. TabRepo cells use a bank of 16,000 configurations, our pool of 4,000 plus 12,000 further draws from the same search spaces fitted on the 182 real datasets only, and are scored against random-K subsets drawn from that bank. The prior ablation uses $2{,}000 \times 2{,}000$ pools with the same protocol.

A.2 HYPERPARAMETER SEARCH SPACES

Every pool is a set of independent random draws from the search space below; no configuration is tuned on any dataset. The same draws are fitted on every synthetic and real task.

LightGBM. We use the LightGBM search space of TabArena (Erickson et al., 2025), shown in Table 4. Training follows the AutoGluon `LGBModel` semantics: up to 10,000 boosting rounds with AutoGluon's adaptive early stopping (offset 20, rate 0.3, maximum patience 300 rounds, scaled down above 10,000 training rows) on the validation split. The pool has $C = 4{,}000$ configurations.

Table 4: The hyperparameter tuning space for LightGBM.

Parameter	Distribution
Learning rate	LogUniform[5e-3, 1e-1]
Feature fraction	Uniform[0.4, 1.0]
Bagging fraction	Uniform[0.7, 1.0]
Bagging frequency	1
# leaves	LogUniformInt[2, 200]
Min. data in leaf	LogUniformInt[1, 64]
Extra trees	{False, True}
Min. data per group	LogUniformInt[2, 100]
Categorical L2	LogUniform[5e-3, 2.0]
Categorical smoothing	LogUniform[1e-3, 100]
Max. categories to one-hot	LogUniformInt[8, 100]
L1 regularization	Uniform[1e-4, 1.0]
L2 regularization	Uniform[1e-4, 2.0]
Max. boosting rounds	10,000
Early stopping	AutoGluon adaptive (offset 20, rate 0.3, max. 300)

LightGBM with TabPrep. The same LightGBM space, with the TabPrep feature generators of Tschalzev et al. (2026) prepended to the pipeline. Following the published protocol, TabPrep is enabled for one configuration in four; an enabled configuration also draws the eight TabPrep axes in Table 5. The pool has $C = 4{,}000$ configurations and shares its LightGBM draws with the plain LightGBM pool, so the two pools differ only in preprocessing.

Table 5: The TabPrep axes drawn for the one-in-four configurations with TabPrep enabled.

Parameter	Distribution
TabPrep enabled	Bernoulli(0.25)
OpenFE max. features	{2000, 1000}
OpenFE generation seed	{42, 84, 168, 336, 672}
Cat. combination max. features	{100, 500}
Cat. combination generation seed	{42, 84, 168, 336, 672}
Random-subset feature combination # subsets	{50, 1, 100}
Random-subset generation seed	{42, 84, 168, 336, 672}
OOF target encoding generation seed	{42, 84, 168, 336, 672}
Group-by max. features	{500, 100, 1000}

MLP with cosine embeddings. The MLP is the independent member model of TabPack (Gorishniy et al., 2026): a feed-forward MLP with cosine numerical embeddings (Gorishniy et al., 2022), trained with the Muon optimizer for the hidden weights and AdamW for the rest. The pool is a frozen registry of $C = 4{,}000$ draws from the space in Table 6. The seven sampled axes follow TabPack's cosine base model; the hidden width is fixed at 384. Members are trained individually with per-member optimizer steps, up to 256 epochs, and early stopping after 16 non-improving validations, without TabPack's online ensemble.

TabPack. TabPack (Gorishniy et al., 2026) trains 64 MLP members as one packed tensor program and combines them with the greedy online ensemble of the TabPack paper. The members are the first 64 configurations of the MLP registry above, so TabPack introduces no hyperparameters of its own. A meta-tuned TabPack is a pack of $K = 64$ configurations selected from the same $4{,}000$ MLP pool with the T+E objective and deployed with the same online ensemble; Figure 1 and Table 3 use this subset.

Table 6: The hyperparameter tuning space for the MLP with cosine embeddings.

Parameter	Distribution
# blocks	UniformInt[1, 4]
Block width	384
Dropout rate	{0.0, Uniform[0.0, 0.5]}
Embedding dim	UniformInt[8, 32], step=4
Embedding init. scale	LogUniform[1e-2, 1e1]
Learning rate (AdamW)	LogUniform[1e-4, 5e-3]
Learning rate (Muon)	LogUniform[1e-3, 1e-1]
Weight decay	LogUniform[1e-3, 1.0]
Batch size	dataset-dependent (TabArena rule)
Max epochs	256
Early-stopping patience	16

A.3 SYNTHETIC PRIORS

All three priors are structural causal models: a directed acyclic graph over latent variables, a random function per edge set, and an observation process that turns node values into a table. This section lists the three parts for each prior in the order TabICLv2, TabICLv1, TabPFNv1, then the sampling settings we used. TabICLv2 is the vendored upstream generator at revision `67eab6b` (PR 111 of `soda-inria/tabicl`); TabICLv1 is tag `v0.1.4` of the same repository; TabPFNv1 is branch `tabpfn_v1` of `automl/TabPFN` at `bef0cf25`.

A.3.1 GRAPH STRUCTURE

TabICLv2. A random Cauchy DAG with $\mathrm{LogInt}(2, 32)$ nodes. The probability of an edge from node i to node j is $\sigma(A + B_i + C_j)$, where A, B_i, C_j are independent standard Cauchy variables; A sets the overall density, B_i and C_j the out- and in-degree of each node. Only edges consistent with a fixed topological order survive. Features and the target are assigned to random nodes; a graph is resampled until the feature nodes and the target node share an ancestor.

TabICLv1. A layered MLP graph as in TabPFNv1: $\mathrm{num_layers}$ layers of width $\mathrm{hidden_dim}$, both drawn from log-scaled truncated normals with per-dataset means in $[1, 6]$ and $[5, 130]$, with block-wise or Bernoulli dropout of edges. In causal mode (sampled per dataset) features and target are read from hidden activations; otherwise the root causes are the features and the last layer is the target.

TabPFNv1. The same layered MLP graph, with layer count and width from gamma meta-distributions (means up to 6 and 300), $\mathrm{num_causes}$ root variables, edge dropout, and the same causal or non-causal read-out. With probability $1/(1 + e^w)$, $w \sim U(2, 10)$, the dataset is instead a draw from an RBF Gaussian process on Gaussian inputs (Hollmann et al., 2023).

A.3.2 EDGE FUNCTIONS

TabICLv2. Eight function types: MLP, tree ensemble (symmetric trees, $\mathrm{LogInt}(1, 128)$ trees of depth 1 to 7), discretization to random centers, Gaussian process with a random kernel, linear, quadratic, EM-style cluster assignment, and product. The first three follow TabPFNv2; the other five are new. With several parents, the prior either concatenates them and applies one function, or applies a function per parent and aggregates by sum, product, max, or log-sum-exp. Each node output is standardized, rescaled by a random feature-importance vector, and multiplied by a random node importance.

TabICLv1. With probability 0.7 an MLP layer: a random affine map followed by an activation drawn from 19 fixed functions (Identity, Tanh, LeakyReLU, ELU, ReLU, ReLU6, SELU, SiLU,

Softplus, Hardtanh, sign, sine, RBF, exponential, $\sqrt{|x|}$, $\mathbf{1}_{|x|\leq 1}$, x^2, $|x|$) or a random Fourier-feature function, the latter ten times as likely; standardization and random rescaling precede each activation, and activations may differ across layers. With probability 0.3 a tree layer: an XGBoost multi-output regressor with at most 4 trees of depth at most 4, fitted on standard-normal targets and evaluated on the parent values.

TabPFNv1. A random affine map followed by one activation per dataset, shared by all layers, drawn from Tanh, Identity, or ReLU as in the released TabPFN code (the paper lists Tanh, LeakyReLU, ELU, and Identity).

A.3.3 Transformations and filters

TabICLv2. No Gaussian noise at nodes; noise enters because only a subset of each node's dimensions is observed. Scalar warping by a Kumaraswamy function after min-max scaling, with exponents in $\mathrm{LogNum}(0.2, 5)$; activation warping off. Categorical features by discretization of sub-vectors, with 2 to 9 categories in our configuration. Outlier clamping at 4 standard deviations, standardization clipped to $[-100, 100]$, constant columns removed, feature order and class labels permuted. Two filters: the ancestor filter on the graph and an ExtraTrees test on the dataset (25 trees, depth 6, bootstrap; reject unless out-of-bag error beats the mean predictor in at least 95% of 200 bootstrap resamples).

TabICLv1. Gaussian noise after every layer with per-dataset standard deviation from $\mathrm{LogNum}(10^{-4}, 0.3)$. Root causes drawn from a normal, uniform, or mixed distribution. With probability 0.2 a dataset has categorical features; each column is converted independently with a uniform probability to $\max(2, \mathrm{round}(\Gamma(1, 10)))$ rank-based categories. Outlier clamping at 4 standard deviations and standardization on features; standardization of the regression target; class labels permuted. Upstream applies no filter; we apply the same ExtraTrees test as for TabICLv2 so that all three corpora are filtered alike.

TabPFNv1. Gaussian noise at every node with per-node mean and standard deviation, the latter from $\mathrm{LogNum}(10^{-4}, 0.3)$. Categorical features as in TabICLv1 (probability 0.2, $\Gamma(1, 10)$ categories, rank-based). Outlier removal and z-normalization of features and target; features rescaled by the fraction of used columns; class boundaries sampled from the data with random label permutation. Upstream applies no filter; we apply the ExtraTrees test.

A.3.4 Sampling settings

Table 7 lists the sampling settings. TabICLv2 is the main prior of Section 3.2. TabICLv1 and TabPFNv1 share one shape specification: dataset i has the same task type, row count, feature count, and class count in both sets, so a difference between them is a difference of generator alone.

A.4 Additional results

Table 8 gives the paired win rates behind Section 4. Table 9 gives the goodness of fit of the scaling forms of Section 5.4. Figures 7–12 repeat the ablations of Section 5 for $K \in \{25, 64, 200\}$. At $K = 25$, TabRepo is ahead at equal budget up to 2^{20} to 2^{22} for all three models, and the synthetic tasks end above its best for all three (Figure 7).

TabRepo overlap. Of the 200 TabRepo datasets we exclude 18. Ten are content-identical to or listed as overlapping with a TabArena dataset: Amazon_employee_access, APSFailure, Bioresponse, blood-transfusion-service-center, diabetes, Diabetes130US, diamonds, hiva_agnostic, jm1, splice. Eight share the source data of a TabArena dataset under a different curation: bank-marketing, churn, credit-g, houses, qsar-biodeg, wine-quality-red, wine-quality-white, wine_quality.

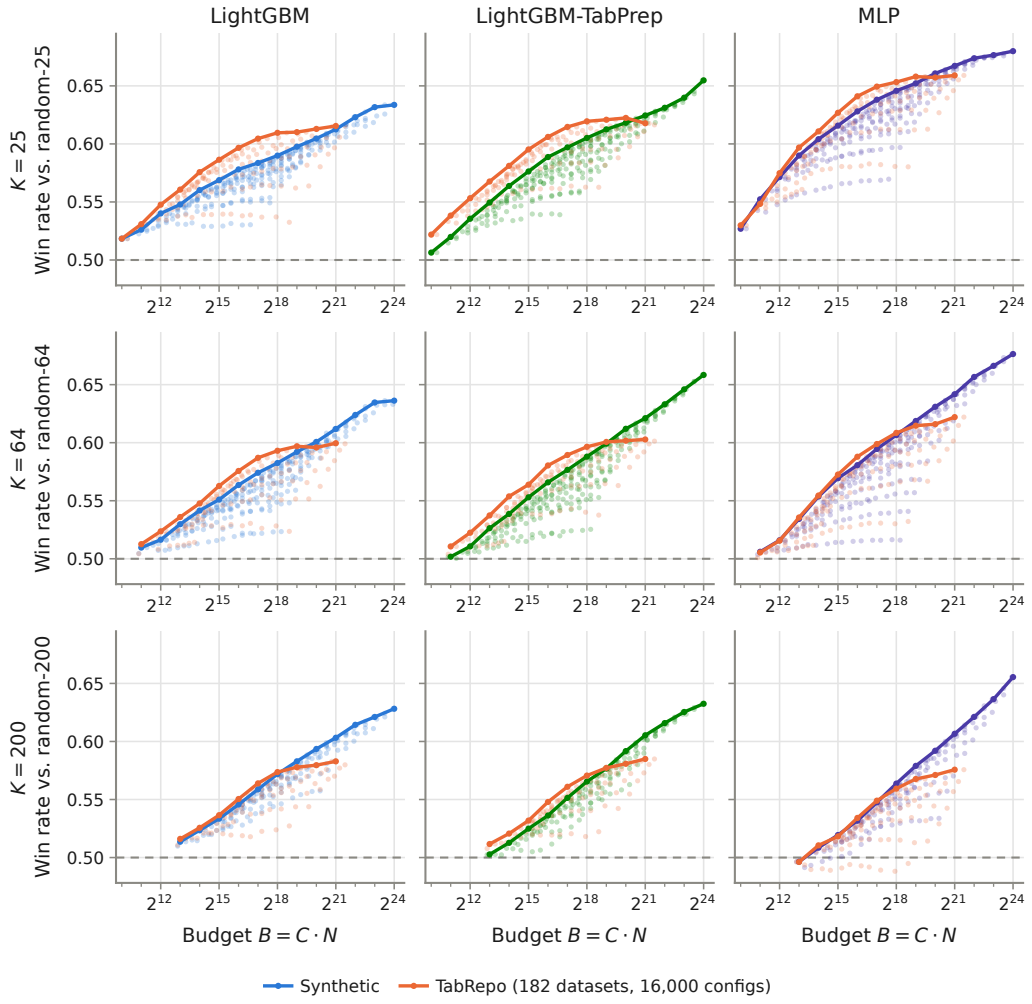


Figure 7: Synthetic versus TabRepo meta-training data at equal budget, as in Figure 3, for $K = 25$, 64, and 200.

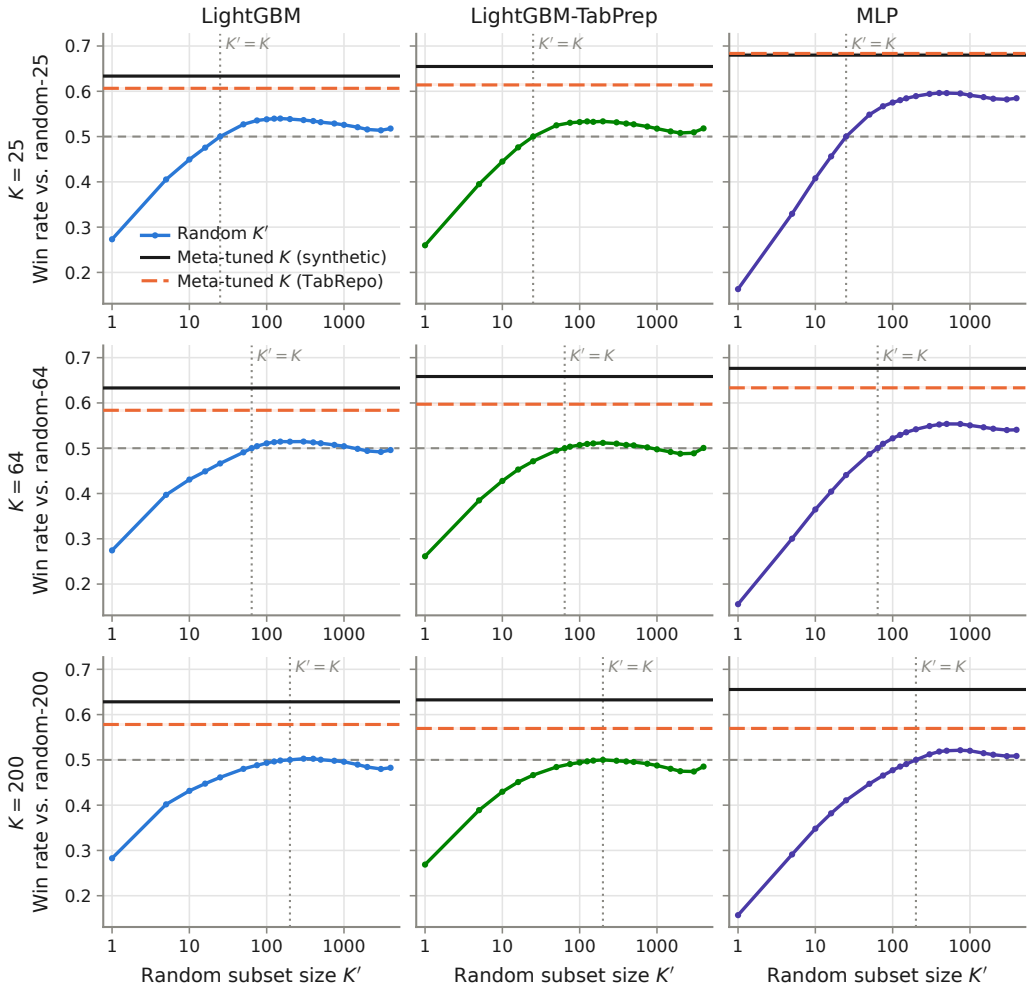


Figure 8: Random subsets of growing size K' against the random-K baseline, as in Figure 4, for $K = 25$, 64, and 200.

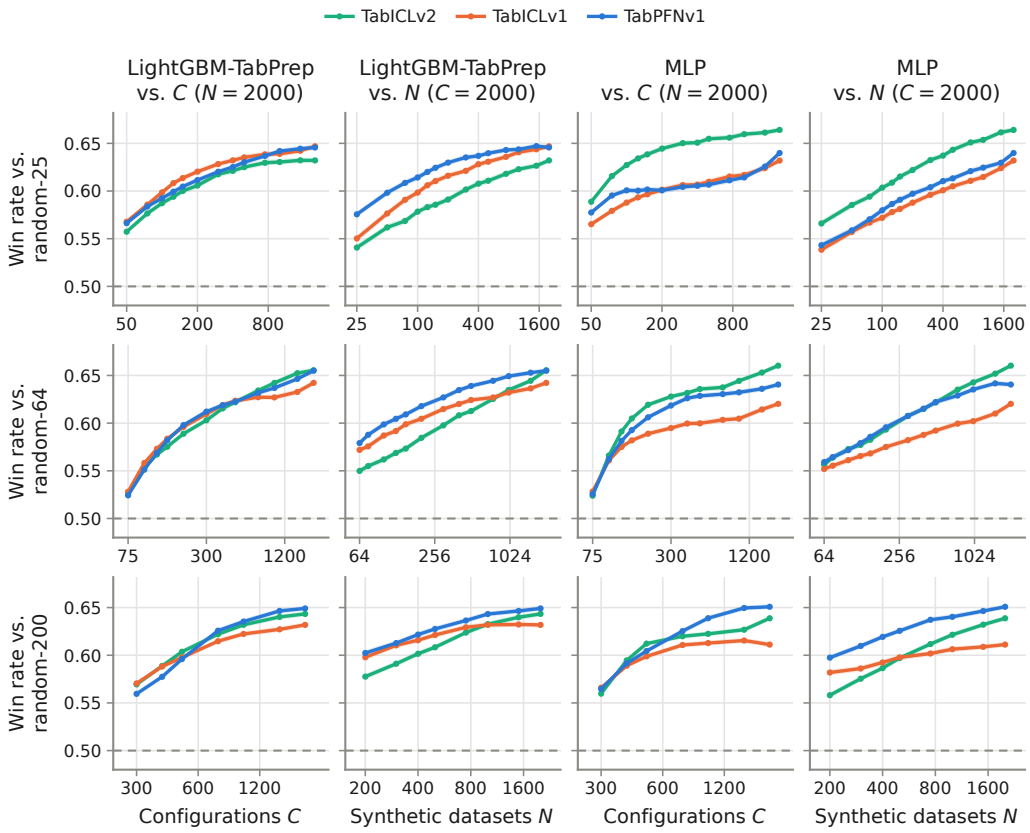


Figure 9: Three priors on $2{,}000 \times 2{,}000$ pools, as in Figure 5, for $K = 25$, 64, and 200.

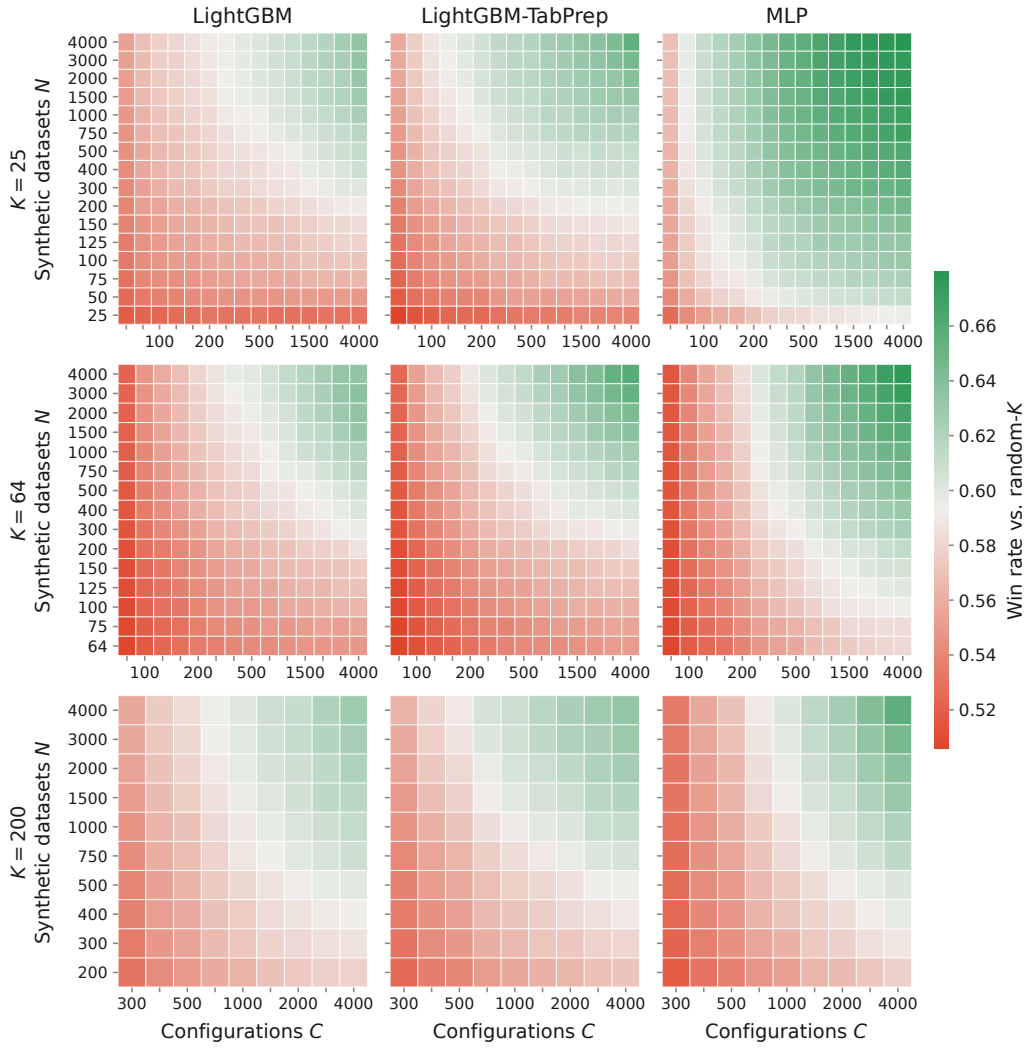


Figure 10: Win rate over the (C, N) grid, as in the top row of Figure 6, for $K = 25$, 64, and 200.

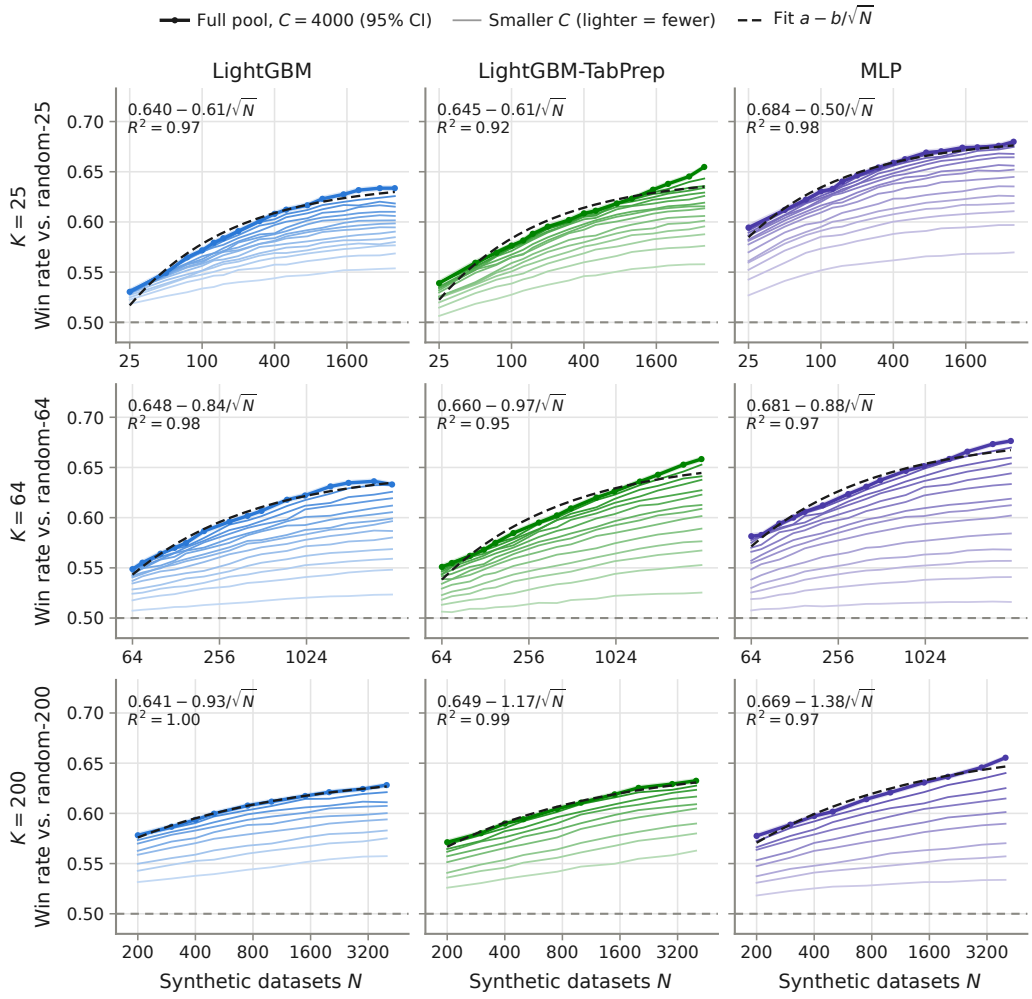


Figure 11: Win rate against the number of synthetic datasets N, one curve per pool size C, with the fit $a - b/\sqrt{N}$ at $C = 4{,}000$, for $K = 25$, 64, and 200.

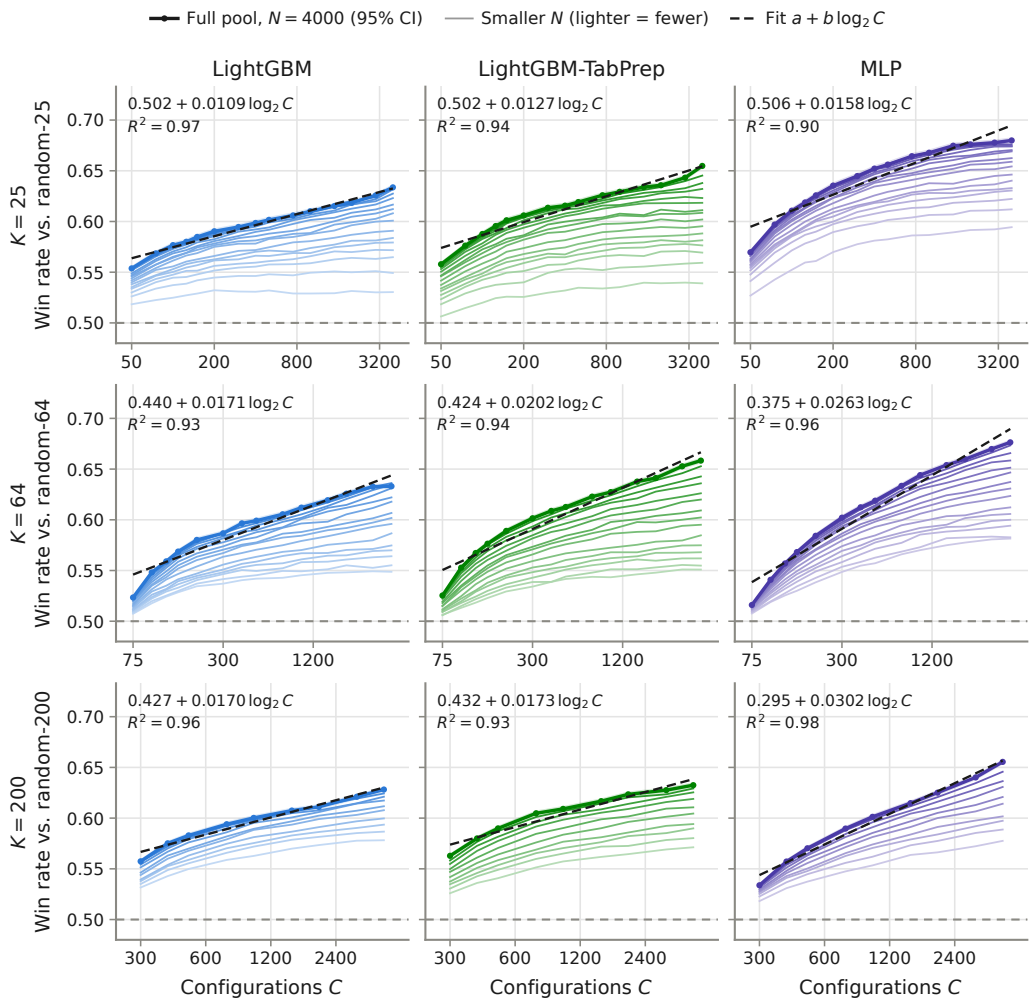


Figure 12: Win rate against the number of configurations C, one curve per number of synthetic datasets N, with the fit $a + b \log_2 C$ at $N = 4{,}000$, for $K = 25$, 64, and 200.

Table 7: Sampling settings of the three synthetic priors. LogU: log-uniform. Splits are train/validation/test and stratified for classification.

Setting	TabICLv2	TabICLv1	TabPFNv1
Generator	`graph_scm`	`mix_scm`	`prior_bag`
Datasets	4,000	2,000	2,000
Task mix (reg/bin/multi)	2,000 / 1,400 / 600	0.4 / 0.3 / 0.3	0.4 / 0.3 / 0.3
Multiclass classes	$U\{3,\dots,10\}$	$U\{3,\dots,10\}$	$U\{3,\dots,10\}$
Rows	LogU$[400, 60{,}000]$	LogU$[1{,}024, 32{,}768]$	LogU$[1{,}024, 32{,}768]$
Features	$U[1, \mathrm{cap}(n)]$, cap $100 \to 20$	LogU$[4, 64]$	LogU$[4, 64]$
Graph nodes	$\mathrm{LogInt}(2, 32)$	MLP, mean layers $[1, 6]$	MLP, gamma layers
Categorical share	$\mathrm{clip}(U(-0.5, 1.2), 0, 1)$	$p = 0.2$ per dataset	$p = 0.2$ per dataset
Node noise	none	$\mathrm{LogNum}(10^{-4}, 0.3)$	$\mathrm{LogNum}(10^{-4}, 0.3)$
Graph filter	ancestor test	none	none
Dataset filter	ExtraTrees, $p < 0.05$	ExtraTrees, $p < 0.05$	ExtraTrees, $p < 0.05$
Split	64 / 16 / 20	64 / 16 / 20	64 / 16 / 20
Attempts per dataset	≤ 100	≤ 64	≤ 64
Base seed	20260804	0	0

Table 8: Paired win rate of each meta-tuned subset against the random subset of its own family (TabPack: its 64 random members), under each deployment rule, on BeyondArena-Lite, by dataset size (half credit for ties)

		Win rate vs. Random subset of the same family							
		Tuned (T)				Tuned + ensembled (T+E)			
Model	Subset	Small	Medium	Large	All	Small	Medium	Large	All
MLP	FT	0.57	0.60	0.46	0.55	0.65	0.50	0.46	0.55
	FTE	0.65	0.52	0.49	0.57	0.66	0.43	0.51	0.55
LightGBM	FT	0.58	0.57	0.58	0.58	0.64	0.74	0.51	0.63
	FTE	0.67	0.61	0.53	0.61	0.61	0.71	0.46	0.60
LightGBM-TabPrep	FT	0.61	0.69	0.47	0.60	0.62	0.69	0.36	0.57
	FTE	0.57	0.57	0.59	0.58	0.66	0.69	0.72	0.68
TabPack	FTE	–	–	–	–	0.57	0.45	0.62	0.55

B THEORY

B.1 SETUP AND MAIN PROPOSITIONS

B.1.1 SETUP

Fix a model f, a learning algorithm $\mathcal{A}$, a candidate pool $\mathcal{C}$ of C hyperparameter configurations, and a deployment budget $1 \leq K \leq C$. A *subset* is $\mathcal{S} \in \Pi_K := \{\mathcal{S} \subseteq \mathcal{C} : |\mathcal{S}| = K\}$. The goal is to choose one subset that performs well on the target distribution of real tasks, using synthetic tasks for meta-training. Performance is win rate against a uniformly random subset of the same size. All statements condition on the candidate pool.

Tasks and deployment. Let $\mathcal{P}$ and $\mathcal{Q}$ be the target (e.g. TabArena or BeyondArena benchmarks (Erickson et al., 2025; Purucker et al., 2026)) and source (e.g. sampled from the TabICLv2 prior (Qu et al., 2026)) distributions over tasks. Sample a target task $\tau \sim \mathcal{P}$, which specifies a row distribution P_τ. Sample the rows of D independently from P_τ and split the dataset into training, validation, and test sets of prescribed sizes. For each configuration, fit

$$\hat{f}_c = \mathcal{A}(f, c, \xi, D),$$

Table 9: Goodness of fit (R^2) of the forms motivated by Appendix B, $a - b/\sqrt{N}$ at the full pool and $a + b \log_2 C$ at the full set of tasks, for the win-rate curves of Figures 11 and 12.

Model	K	$a - b/\sqrt{N}$ ($C = 4{,}000$)	$a + b \log_2 C$ ($N = 4{,}000$)
LightGBM	25	0.97	0.97
	64	0.98	0.93
	200	1.00	0.96
LightGBM-TabPrep	25	0.92	0.94
	64	0.95	0.94
	200	0.99	0.93
MLP	25	0.98	0.90
	64	0.97	0.96
	200	0.97	0.98

where ξ collects the randomness in the dataset split and in the training of every configuration (initialization, subsampling, and any other stochastic step of $\mathcal{A}$). Each configuration is fitted once per task, and every subset that contains it, including the random reference it is compared against, reuses that fit. A *deployment rule* $\mathcal{E}$ takes as an input a subset (its fitted members, train and validation subsets) and outputs a single predictor. We then score it on the test set:

$$L(\mathcal{S}; D, \hat{f}, \xi) = \ell\Big(\mathcal{E}(\mathcal{S}; D, \hat{f}, \xi); D_{\text{test}}\Big). \tag{1}$$

We require that $\mathcal{E}$ be deterministic given its inputs and that the same $\mathcal{E}$ be applied to a subset and to any subset it is compared against.

Two deployment rules. We use the two regimes of Erickson et al. (2025); Purucker et al. (2026). *Tuned* (T) keeps the validation-best member,

$$\mathcal{E}_{\text{T}}(\mathcal{S}) = \hat{f}_{c^*}, \qquad c^* = \arg\min_{c \in \mathcal{S}} \ell(\hat{f}_c; D_{\text{val}}).$$

Tuned and ensembled (T+E) runs B steps of greedy ensemble selection with replacement on the validation split (Caruana et al., 2004). Writing w_c for the share of the B slots taken by configuration c,

$$\mathcal{E}_{\text{T+E}}(\mathcal{S}) = \sum_{c \in \mathcal{S}} w_c \, \hat{f}_c, \qquad w_c \geq 0, \quad \sum_{c \in \mathcal{S}} w_c = 1.$$

The tuned regime is the special case of tuned and ensembled that puts all weight on one member.

Population objective. Let $h(a, b) = \mathbf{1}_{a<b} + \frac{1}{2}\mathbf{1}_{a=b}$. The per-task utility of a subset is

$$u(\mathcal{S}, \tau) = \mathbb{E}_{D, \hat{f}, \xi | \tau, \, \mathcal{R} \sim \text{Unif}(\Pi_K)} \left[h\Big(L(\mathcal{S}; D, \hat{f}, \xi), L(\mathcal{R}; D, \hat{f}, \xi) \Big) \right], \tag{2}$$

and the population objective is

$$U_{\mathcal{P}}(\mathcal{S}) = \mathbb{E}_{\tau \sim \mathcal{P}}[u(\mathcal{S}, \tau)], \qquad \mathcal{S}^*_{\mathcal{P}} \in \arg\max_{\mathcal{S} \in \Pi_K} U_{\mathcal{P}}(\mathcal{S}). \tag{3}$$

A uniformly random subset scores $\mathbb{E}_{\mathcal{S} \sim \text{Unif}(\Pi_K)}[U_{\mathcal{P}}(\mathcal{S})] = 1/2$ by exchangeability (proof in §B.2), so $U_{\mathcal{P}}(\mathcal{S}) - 1/2$ is the gain over random search at the same budget.

Evaluations and statistics. Given a task τ, let $\omega = (D, \hat{f}, \xi)$ be the *evaluation*, with law μ_τ on a space Ω, and write $L(\mathcal{S}; \omega)$ for the deployed loss equation 1. The deployment rule is deterministic given its inputs, so once ω is fixed the deployed loss of every subset is fixed: $L(\mathcal{S}; \cdot)$ is a map on Ω, which we assume measurable, as the expectation in equation 2 already requires. Write $\bar{\mathcal{P}} = \mathbb{E}_{\tau \sim \mathcal{P}}[\mu_\tau]$ and $\bar{\mathcal{Q}} = \mathbb{E}_{\tau \sim \mathcal{Q}}[\mu_\tau]$ for the marginal laws of ω. A *statistic* is a measurable map $\Phi : \Omega \to \mathcal{X}$, and $\mathcal{P}_\Phi = \bar{\mathcal{P}} \circ \Phi^{-1}$, $\mathcal{Q}_\Phi = \bar{\mathcal{Q}} \circ \Phi^{-1}$ are the laws of $\Phi(\omega)$. In this notation, equation 2–equation 3 read

$$U_{\mathcal{P}}(\mathcal{S}) = \mathbb{E}_{\tau \sim \mathcal{P}} \, \mathbb{E}_{\omega \sim \mu_\tau} \, \mathbb{E}_{\mathcal{R}}[h(L(\mathcal{S}; \omega), L(\mathcal{R}; \omega))], \qquad \mathcal{R} \sim \text{Unif}(\Pi_K) \text{ independent of } (\tau, \omega).$$

B.1.2 UPPER BOUND ON THE IMPROVEMENT

Suppose we have access to the target distribution $\mathcal{P}$ itself, so $U_{\mathcal{P}}$ is known on all of Π_K and selection carries no estimation error. We ask how many of the subsets in Π_K improve upon the best subset seen so far. Suppose the $M = \binom{C}{K}$ subsets are revealed in uniformly random order. A subset improves the oracle's choice precisely when its utility sets a new record.

Proposition 1 (Oracle record count) *If the oracle utilities are distinct, the ith revealed subset sets a record with probability $1/i$ over the uniformly random reveal order. Counting the initial subset as a record,*

$$\mathbb{E}[\textit{number of records among the } M \textit{ subsets}] = \sum_{i=1}^{M} \frac{1}{i} = \log M + O(1).$$

The count is a sum of independent Bernoulli variables with mean $\log M + O(1)$ (Rényi, 1962): improvements arrive at rate $1/i$ and thin out as the search proceeds. Since $\log M = K \log(C/K) + O(K)$, at fixed $K \ll C$ doubling the pool adds about $K \log 2$ of them. Meta-tuning thus is the problem of recovering enough of the $\log M$ improvements, having access only to the N source tasks.

B.1.3 TRANSFER FROM SYNTHETIC TO REAL TASKS

The target distribution is not available for meta-tuning: we observe only a benchmark of real tasks, which we reserve for evaluation. We therefore select the subset on a source distribution $\mathcal{Q}$ of synthetic tasks and transfer it. We sample N independent source tasks $\tau_i \sim \mathcal{Q}$ and one evaluation $(D_i, \hat{f}_i, \xi_i)$ for each. Write $L_i(\mathcal{S}) = L(\mathcal{S}; D_i, \hat{f}_i, \xi_i)$. The empirical counterpart of equation 3 is

$$\widehat{U}_N(\mathcal{S}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathcal{R} \sim \mathrm{Unif}(\Pi_K)} \left[h(L_i(\mathcal{S}), L_i(\mathcal{R})) \right], \tag{4}$$

and $\mathbb{E}[\widehat{U}_N(\mathcal{S})] = U_{\mathcal{Q}}(\mathcal{S})$ for every fixed subset. Its expectation is $U_{\mathcal{Q}}$, not $U_{\mathcal{P}}$; we quantify the price of this approximation.

Assumption 1 (Comparison fidelity) *Source and target subset comparisons satisfy*

$$\sup_{\mathcal{S}, \mathcal{T} \in \Pi_K} |[U_{\mathcal{P}}(\mathcal{S}) - U_{\mathcal{P}}(\mathcal{T})] - [U_{\mathcal{Q}}(\mathcal{S}) - U_{\mathcal{Q}}(\mathcal{T})]| \le \varepsilon. \tag{5}$$

Proposition 2 (Transfer of subset selection) *Let $\widehat{\mathcal{S}}$ be the subset returned by the selector and*

$$\eta_N = \max_{\mathcal{S} \in \Pi_K} \widehat{U}_N(\mathcal{S}) - \widehat{U}_N(\widehat{\mathcal{S}})$$

its suboptimality on the source tasks. Under Assumption 1, with probability at least $1 - \delta$,

$$U_{\mathcal{P}}(\mathcal{S}^*_{\mathcal{P}}) - U_{\mathcal{P}}(\widehat{\mathcal{S}}) \le \varepsilon + 2\sqrt{\frac{\log(2\binom{C}{K}/\delta)}{2N}} + \eta_N. \tag{6}$$

The bound separates selection error, source-target mismatch, and finite-task uncertainty. The number of source tasks reduces the second term, of order $\sqrt{\log M / N}$, but cannot remove systematic reversals of subset preferences (ε) or the error of the selection algorithm (η_N). We next investigate when Assumption 1 holds.

Lemma 1 (Comparison discrepancy through a statistic) *The comparison* factors through Φ *if for every $(\mathcal{S}, \mathcal{R}) \in \Pi_K \times \Pi_K$ there is a measurable $g_{\mathcal{S},\mathcal{R}} : \mathcal{X} \to [0, 1]$, depending on $(\mathcal{S}, \mathcal{R})$ only, such that*

$$h(L(\mathcal{S}; \omega), L(\mathcal{R}; \omega)) = g_{\mathcal{S},\mathcal{R}}(\Phi(\omega)) \qquad \textit{for every } \omega \in \Omega. \tag{7}$$

If the comparison factors through Φ, then Assumption 1 holds with $\varepsilon = 2\,\mathrm{TV}(\mathcal{P}_\Phi, \mathcal{Q}_\Phi)$, where $\mathrm{TV}(P, Q) = \sup_A |P(A) - Q(A)|$. If moreover $\Phi = \kappa \circ \Psi$ for a statistic $\Psi : \Omega \to \mathcal{Y}$ and a measurable $\kappa : \mathcal{Y} \to \mathcal{X}$, then the comparison also factors through Ψ, and $\mathrm{TV}(\mathcal{P}_\Phi, \mathcal{Q}_\Phi) \le \mathrm{TV}(\mathcal{P}_\Psi, \mathcal{Q}_\Psi)$.

Proposition 3 (Rank sufficiency) *Let $\Phi(\omega)$ be the ordering of the M losses $\{L(\mathcal{S};\omega)\}_{\mathcal{S}\in\Pi_K}$, ties recorded. Then, for any deployment rule, Assumption 1 holds with $\varepsilon = 2\,\mathrm{TV}(\mathcal{P}_\Phi, \mathcal{Q}_\Phi)$. Under $\mathcal{E}_\mathrm{T}$ the same holds for the finer statistic $\Psi(\omega)$, the validation and test orderings of the pool $\mathcal{C}$.*

Hence what must transfer is the ranking of subsets by deployed loss. Under T this reduces to the validation and test ordering of individual configurations. Whether the synthetic prior preserves these orderings is an empirical question.

B.1.4 Analysis of the forward-selection heuristic

The inner expectation in equation 4 runs over $\binom{C}{K}$ references, so in practice we draw R independent references $\mathcal{R}_r \sim \mathrm{Unif}(\Pi_K)$ and optimize

$$\widetilde{U}_{N,R}(\mathcal{S}) = \frac{1}{NR}\sum_{i=1}^{N}\sum_{r=1}^{R} h(L_i(\mathcal{S}), L_i(\mathcal{R}_r))\,. \tag{8}$$

Proposition 4 (Reference averaging) *Write $\mathbb{E}_\mathcal{R}$ and $\mathrm{Var}_\mathcal{R}$ for the average over the R reference draws with the source tasks held fixed. For every fixed subset,*

$$\mathbb{E}_\mathcal{R}\Big[\widetilde{U}_{N,R}(\mathcal{S})\Big] = \widehat{U}_N(\mathcal{S}), \qquad \mathrm{Var}_\mathcal{R}\Big(\widetilde{U}_{N,R}(\mathcal{S})\Big) \le \frac{1}{4R}. \tag{9}$$

References thus add no bias, and their sampling noise vanishes as $1/R$.

Forward selection. Exhaustive search over $\binom{C}{K}$ subsets is impractical. Starting from $\mathcal{S}_0 = \emptyset$, we run

$$c_k \in \operatorname*{arg\,max}_{c\in\mathcal{C}\setminus\mathcal{S}_{k-1}} \widetilde{U}_{N,R}(\mathcal{S}_{k-1}\cup\{c\}),$$
$$\mathcal{S}_k = \mathcal{S}_{k-1}\cup\{c_k\}, \qquad k = 1,\dots,K.$$

We return exactly K configurations, even if an addition lowers the objective. The objective is not monotone in general, so forward selection is a heuristic with no generic greedy-optimality guarantee. In Proposition 2, both the gap between $\widetilde{U}_{N,R}$ and $\widehat{U}_N$ and the greedy suboptimality enter through η_N.

B.2 Proofs

Proof of the random-K baseline. Let $\mathcal{S},\mathcal{R}$ be i.i.d. uniform on Π_K, independent of (τ,ω), and write $a = L(\mathcal{S};\omega)$, $b = L(\mathcal{R};\omega)$, so that $\mathbb{E}_\mathcal{S}[U_\mathcal{P}(\mathcal{S})] = \mathbb{E}_{\tau,\omega}\,\mathbb{E}_{\mathcal{S},\mathcal{R}}[h(a,b)]$. For fixed (τ,ω), the pairs $(\mathcal{S},\mathcal{R})$ and $(\mathcal{R},\mathcal{S})$ have the same law, so $\mathbb{E}_{\mathcal{S},\mathcal{R}}[h(a,b)] = \mathbb{E}_{\mathcal{S},\mathcal{R}}[h(b,a)]$, and adding the two with $h(a,b)+h(b,a)=1$ gives $\mathbb{E}_{\mathcal{S},\mathcal{R}}[h(a,b)] = \frac{1}{2}$ for every (τ,ω), hence $\mathbb{E}_\mathcal{S}[U_\mathcal{P}(\mathcal{S})] = \frac{1}{2}$. □

Proof of Proposition 1. Write $M = \binom{C}{K}$ and let $U_i = U_\mathcal{P}(\mathcal{S}_i)$ be the utility of the ith revealed subset. Let ρ_i be the rank of U_i, from largest to smallest, among $U_1,\dots,U_i$, so that $I_i = \mathbf{1}_{\rho_i=1}$ indicates a record. We first show that $\rho_1,\dots,\rho_M$ are independent with ρ_i uniform on $\{1,\dots,i\}$; this is the classical record theorem of Rényi (1962).

Let π be the permutation that sorts $(U_1,\dots,U_M)$. Because the reveal order is uniform and the utilities are distinct, every ordering is equally likely, so π is uniform over the $M!$ permutations. The rank vector $(\rho_1,\dots,\rho_M)$ is a function of π, and this function is a bijection onto $\{1\}\times\{1,2\}\times\cdots\times\{1,\dots,M\}$: it is injective because π is recovered by inserting U_i at position ρ_i among $U_1,\dots,U_{i-1}$ for $i = 1,\dots,M$, and both sets have $M!$ elements. Hence $(\rho_1,\dots,\rho_M)$ is uniform on that product set, and for each marginal we can calculate

$$\Pr(\rho_i = k_i) = \sum_{k_1=1}^{1}\sum_{k_2=1}^{2}\cdots\sum_{k_{i-1}=1}^{i-1}\sum_{k_{i+1}=1}^{i+1}\cdots\sum_{k_M=1}^{M}\frac{1}{M!} = \frac{M!}{i}\cdot\frac{1}{M!} = \frac{1}{i},$$

since the retained sums have $1\cdot 2\cdots(i-1)\cdot(i+1)\cdots M = M!/i$ terms. For independence, we observe that $\Pr(\rho_1=k_1,\dots,\rho_M=k_M) = 1/M! = \Pr(\rho_1=k_1)\cdots\Pr(\rho_M=k_M)$.

Therefore we also have that $\Pr(I_i = 1) = \Pr(\rho_i = 1) = 1/i$, and $I_1, \dots, I_M$ are independent because each is a function of a different ρ_i. By linearity, $\mathbb{E}\sum_i I_i = \sum_i \Pr(I_i = 1) = H_M$, and comparing the sum with $\int x^{-1}dx$ gives $\log(M+1) \le H_M \le 1+\log M$. Finally $(C/K)^K \le M \le (C/K)^K e^K$ gives $\log M = K\log(C/K) + O(K)$. □

Proof of Proposition 2. Fix $\mathcal{S} \in \Pi_K$. By equation 4, $\hat{U}_N(\mathcal{S})$ is the mean of N i.i.d. terms in $[0,1]$ with mean $U_{\mathcal{Q}}(\mathcal{S})$. Hoeffding's inequality gives $\Pr(|\hat{U}_N(\mathcal{S}) - U_{\mathcal{Q}}(\mathcal{S})| > t) \le 2\exp(-2Nt^2)$, and a union bound over the $\binom{C}{K}$ subsets gives

$$\Pr\Big(\sup_{\mathcal{S}\in\Pi_K} |\hat{U}_N(\mathcal{S}) - U_{\mathcal{Q}}(\mathcal{S})| > t\Big) \le 2\binom{C}{K}\exp(-2Nt^2). \tag{10}$$

Setting the right-hand side to δ and solving for t yields $t_N = \sqrt{\log(2\binom{C}{K}/\delta)/(2N)}$. On the complementary event, which has probability at least $1-\delta$, let $\mathcal{S}^*$ maximize $U_{\mathcal{P}}$. Then

$$\begin{aligned} U_{\mathcal{P}}(\mathcal{S}^*) - U_{\mathcal{P}}(\hat{\mathcal{S}}) &\le U_{\mathcal{Q}}(\mathcal{S}^*) - U_{\mathcal{Q}}(\hat{\mathcal{S}}) + \varepsilon && \text{by equation 5} \\ &\le \hat{U}_N(\mathcal{S}^*) - \hat{U}_N(\hat{\mathcal{S}}) + \varepsilon + 2t_N && \text{by uniform concentration} \\ &\le \eta_N + \varepsilon + 2t_N && \text{by the definition of } \eta_N. \end{aligned}$$

The bound holds for any selector. Ours maximizes the Monte Carlo objective $\widetilde{U}_{N,R}$ rather than $\widehat{U}_N$, and both that gap and the greedy heuristic's are part of η_N. □

Proof of Lemma 1. Since $\mathcal{R} \sim \mathrm{Unif}(\Pi_K)$ is independent of (τ, ω) and all terms are bounded, the objective equation 3 and the factoring equation 7 give

$$\begin{aligned} U_{\mathcal{P}}(\mathcal{S}) &= \mathbb{E}_{\tau\sim\mathcal{P}}\,\mathbb{E}_{\omega\sim\mu_\tau}\,\mathbb{E}_{\mathcal{R}}[h(L(\mathcal{S};\omega), L(\mathcal{R};\omega))] \\ &= \mathbb{E}_{\mathcal{R}}\,\mathbb{E}_{\tau\sim\mathcal{P}}\,\mathbb{E}_{\omega\sim\mu_\tau}[g_{\mathcal{S},\mathcal{R}}(\Phi(\omega))] = \mathbb{E}_{\mathcal{R}}\,\mathbb{E}_{\mathcal{P}_\Phi}[g_{\mathcal{S},\mathcal{R}}], \end{aligned} \tag{11}$$

where the last equality uses that $\bar{\mathcal{P}} = \mathbb{E}_{\tau\sim\mathcal{P}}[\mu_\tau]$ is the marginal law of ω and $\mathcal{P}_\Phi$ is the law of $\Phi(\omega)$ under it; likewise $U_{\mathcal{Q}}(\mathcal{S}) = \mathbb{E}_{\mathcal{R}}\,\mathbb{E}_{\mathcal{Q}_\Phi}[g_{\mathcal{S},\mathcal{R}}]$. For any measurable $g : \mathcal{X} \to [0,1]$, the layer-cake formula gives $|\mathbb{E}_P g - \mathbb{E}_Q g| = |\int_0^1 (P(g > t) - Q(g > t))\,dt| \le \mathrm{TV}(P, Q)$. Applying it to each $g_{\mathcal{S},\mathcal{R}}$ and averaging over $\mathcal{R}$ gives $|U_{\mathcal{P}}(\mathcal{S}) - U_{\mathcal{Q}}(\mathcal{S})| \le \mathrm{TV}(\mathcal{P}_\Phi, \mathcal{Q}_\Phi)$ for every $\mathcal{S}$. Applying this to $\mathcal{S}$ and $\mathcal{T}$ and using the triangle inequality yields equation 5 with $\varepsilon = 2\,\mathrm{TV}(\mathcal{P}_\Phi, \mathcal{Q}_\Phi)$.

If $\Phi = \kappa \circ \Psi$, then equation 7 holds with Ψ and $g_{\mathcal{S},\mathcal{R}} \circ \kappa$ in place of Φ and $g_{\mathcal{S},\mathcal{R}}$. Since $\{\Phi \in A\} = \{\Psi \in \kappa^{-1}(A)\}$, $|\mathcal{P}_\Phi(A) - \mathcal{Q}_\Phi(A)| = |\mathcal{P}_\Psi(\kappa^{-1}(A)) - \mathcal{Q}_\Psi(\kappa^{-1}(A))| \le \mathrm{TV}(\mathcal{P}_\Psi, \mathcal{Q}_\Psi)$ for every A. □

Proof of Proposition 3. Let $\Phi(\omega) = (\mathrm{rank}_\omega(\mathcal{S}))_{\mathcal{S}\in\Pi_K}$ be the vector of ranks of the losses $\{L(\mathcal{S};\omega)\}_{\mathcal{S}\in\Pi_K}$, equal losses receiving equal rank; it takes values in a finite set. Since $h(a,b)$ depends only on whether $a < b$, $a = b$, or $a > b$, and dense ranks preserve this, $h(L(\mathcal{S};\omega), L(\mathcal{R};\omega)) = \mathbf{1}_{\mathrm{rank}_\omega(\mathcal{S})<\mathrm{rank}_\omega(\mathcal{R})} + \frac{1}{2}\mathbf{1}_{\mathrm{rank}_\omega(\mathcal{S})=\mathrm{rank}_\omega(\mathcal{R})}$, which is equation 7. Lemma 1 gives $\varepsilon = 2\,\mathrm{TV}(\mathcal{P}_\Phi, \mathcal{Q}_\Phi)$.

Under $\mathcal{E}_{\mathrm{T}}$, let $\Psi(\omega)$ be the pair of dense-rank vectors of the pool $\mathcal{C}$ under the validation losses and under the test losses. The deployed member of $\mathcal{S}$ is its validation-rank minimizer, ties broken by a fixed rule, and the deployed losses of two subsets compare as the test ranks of their deployed members, so $\Phi = \kappa \circ \Psi$ for a fixed κ. The second part of Lemma 1 applies. □

Proof of Proposition 4. Fix $\mathcal{S}$, hold the source tasks fixed, and set $Z_r = \frac{1}{N}\sum_{i=1}^N h(L_i(\mathcal{S}), L_i(\mathcal{R}_r))$, so that $\widetilde{U}_{N,R}(\mathcal{S}) = R^{-1}\sum_r Z_r$. The Z_r are i.i.d. functions of the independent draws $\mathcal{R}_r$, lie in $[0,1]$, and have mean $\widehat{U}_N(\mathcal{S})$ by equation 4. Hence $\mathbb{E}_{\mathcal{R}}[\widetilde{U}_{N,R}(\mathcal{S})] = \widehat{U}_N(\mathcal{S})$, and $\mathrm{Var}_{\mathcal{R}}(\widetilde{U}_{N,R}(\mathcal{S})) = R^{-1}\,\mathrm{Var}(Z_1) \le 1/(4R)$, since a variable supported on $[0,1]$ has variance at most $1/4$. □