UNDERSTANDING CONTEXT QUALITY AND IMPROVING BATCH RETRIEVAL FOR TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models predict new samples from labeled reference examples without task-specific model training. Using a separate context for each query or query cluster incurs substantial computational cost, motivating the selection of an effective shared context for an entire query batch under a limited budget. Existing retrieval methods typically use fixed selection rules whose performance varies substantially across tasks. Through exhaustive synthetic experiments and controlled analyses on real datasets, we find that a sample's predictive contribution is closely related to local class support, support accumulation, allocation across queries, and class composition. Based on these observations, we propose Preference-Aware Integrated Retrieval (PAIR), an adjustable retrieval framework that combines support accumulation, coverage allocation across queries, and class reservation, with configurations selected through validation. Its coverage objective evaluates candidates according to existing coverage and exhibits diminishing returns, enabling efficient greedy optimization with an approximation guarantee. Across 51 classification tasks, PAIR achieves the best mean rank of 1.91 among seven methods, with its untuned default ranking second, and the gains persist across four frozen models, multiple context budgets, and regression tasks. Its mean inference time is approximately one twenty-fifth that of per-query retrieval, with corresponding differences in predictive quality.

1 INTRODUCTION

In applications such as clinical risk assessment (Rajkomar et al., 2018), network traffic detection (Luay et al., 2025), and industrial quality control (Wuest et al., 2016), prediction systems use existing labeled records to assess incoming batches of new samples under requirements on predictive quality, response time, and computation. Each new task requires its own model selection, tuning, and training, and these steps must be repeated whenever the data or requirements change (Feurer et al., 2015; Erickson et al., 2025).

Tabular foundation models (TFMs) offer a new prediction workflow for this setting. Models such as TabPFN and TabICL acquire general predictive capabilities through extensive pretraining tasks (Hollmann et al., 2023; 2025; Qu et al., 2025). At deployment, they receive labeled reference examples and, with their parameters frozen, make predictions for new samples. However, the number of reference examples a TFM can accommodate is limited by model capacity and inference cost (Zeng et al., 2025), requiring the system to select a compact context from the available labeled data. Existing retrieval methods construct contexts for individual queries (Thomas et al., 2024) or query clusters (Xu et al., 2025; Heredge et al., 2026), each requiring a separate model call, so that inference time grows with the number of clusters. Most TFMs natively predict a full query batch from one shared context in a single call, yet retrieval designed for this batch setting remains largely unexplored. We study *fixed-batch context retrieval*: given a query batch and a context budget, select one subset of the labeled pool for the entire batch to share.

Sharing a context turns retrieval into an allocation problem under a limited budget. Queries have diverse reference needs but share every budget position, creating competition for limited capacity. A candidate's marginal contribution also depends on the examples already selected. Understanding

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.

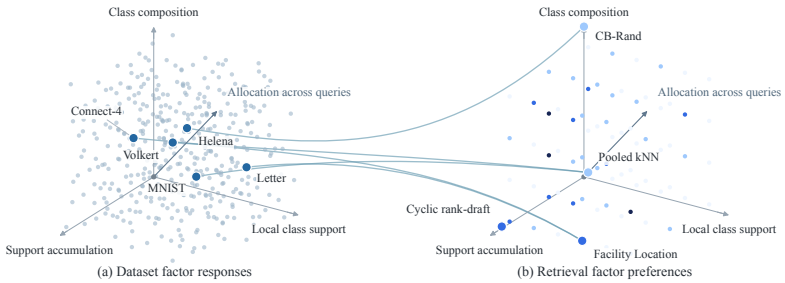

Figure 1: Dataset factor responses and retrieval preferences. (a) Five datasets positioned by normalized absolute responses to the four factors, with pale points illustrating the space. (b) Retrieval configurations in a schematic layout, darker points marking more frequent selection by validation across 51 tasks. Curves connect each dataset to its lowest-loss baseline among the five of Table 1.

shared retrieval therefore requires answering two questions: which factors influence a sample's contribution to prediction, and how do these contributions change with the composition of the context?

To answer these questions, we compare prediction changes induced by sample additions and exchanges in a synthetic task that permits exhaustive context evaluation, then examine the same properties in controlled experiments on real data. We identify four properties that govern context quality: (i) *local class support*: examples from the query's class tend to provide greater predictive benefit when they are closer to the query. (ii) *support accumulation*: marginal gains diminish with accumulated support, so that the same candidate yields different gains in different contexts. (iii) *allocation across queries*: under a shared budget, strengthening one query's support can come at the cost of another's. (iv) *class composition*: the context's class composition shifts predictions even with local support held constant. The strength of these effects varies across tasks, and retrieval rules place different emphasis on the factors. Figure 1 summarizes these differences: the best-performing baseline changes across the five diagnostic datasets. This motivates a common retrieval framework that combines these choices and adjusts them to task requirements.

Based on this analysis, we propose Preference-Aware Integrated Retrieval (PAIR), which values a candidate's contribution according to the coverage each query already receives. Its three adjustable choices, namely how coverage accumulates over multiple neighbors, how additional coverage is valued across queries, and how much of the budget is reserved across classes, define interpretable retrieval preferences selected through validation. The coverage objective is monotone with diminishing returns, and greedy extension from the class-reserved set carries an approximation guarantee, enabling efficient construction.

We assess the framework through benchmark comparisons, an untuned default configuration, and ablations that restrict the available configuration choices. Across 51 classification tasks at a context budget of 100, PAIR achieves the best mean rank of 1.91 among seven methods with its untuned default second, and the gains persist across four frozen TFMs, further context budgets, and regression tasks. End-to-end inference on five tasks takes approximately one twenty-fifth the time of per-query kNN, with task-dependent changes in predictive quality.

Our core contributions are:

- **An analysis of shared-context quality under fixed query batches.** Through exhaustive synthetic comparisons and controlled real-data experiments, we examine how sample contributions relate to local class support, support accumulation, allocation across queries, and class composition. These findings provide a basis for interpreting the choices made by existing selectors across tasks.
- **Preference-Aware Integrated Retrieval.** We introduce an adjustable, unified selection framework that combines support accumulation, coverage allocation across queries, and class reservation. The framework dynamically evaluates candidates according to existing coverage and provides approximation guarantees for greedy optimization of the coverage objective.

2 Related work

Tabular foundation models under context limits. Prior-fitted networks learn an inference procedure from synthetic tasks and apply it to new datasets through labeled context (Müller et al., 2022). TabPFN established this paradigm, and subsequent TFMs improved predictive accuracy and expanded the supported context size (Hollmann et al., 2023; 2025; Grinsztajn et al., 2025; 2026; Qu et al., 2025; Zeng et al., 2025). Mechanistic analyses identify attention-weighted label voting and class-conditional readouts in these models (Biloš et al., 2026). Related studies describe TFM generalization as retrieval and aggregation or make this view explicit through in-context kernel regression (Shaheen et al., 2026; Miftachov et al., 2026). These findings make the composition of the labeled context central to TFM inference. Architectural developments enlarge the supported context, leaving finite context and computation budgets as practical constraints.

Context construction and retrieval for TFMs. Methods for bounded contexts follow several routes. TuneTables learns a compact continuous context (Feuer et al., 2024), and in-context data distillation optimizes synthetic rows (Ma et al., 2024). Retrieval-based methods preserve observed training rows. LoCalPFN combines per-query nearest-neighbor retrieval with task-specific fine-tuning (Thomas et al., 2024). MixturePFN routes queries among multiple prompters and adapts the model to the downstream dataset (Xu et al., 2025). CRUMB clusters test queries and selects a distribution-matched training subset for each cluster (Heredge et al., 2026). VIP-COp uses model-aware validation feedback to optimize a context shared across test samples (Chen et al., 2026). Fixed-batch shared retrieval considers one incoming query batch as the conditioning unit: the complete batch jointly determines one context under a fixed row budget, and the predictor remains frozen. CRUMB instantiates this regime through its single-cluster configuration. VIP-COp provides a task-level shared-context comparison whose selection is independent of the current query batch.

3 Understanding Context Quality under a Shared Budget

3.1 Problem setting

Let $\mathcal{D} = \{(x_j, y_j)\}_{j=1}^{N}$ be a labeled pool, $\mathcal{Q} = \{q_i\}_{i=1}^{M}$ a fixed query batch, and B the shared context budget. A fixed-batch selector chooses $S = A(\mathcal{D}, \mathcal{Q}, B) \subseteq \mathcal{D}$ with $|S| = B$, and a frozen TFM predicts the complete batch in one call, $f(\mathcal{Q} \mid S)$. Selection uses query features and labeled pool rows. Each chosen row consumes shared capacity and can affect many queries. We investigate which context properties accompany these effects by examining row additions and exchanges, checking the patterns on real data, and describing existing selectors through the same properties, using query labels only in diagnostic analyses and evaluation.

3.2 From row contributions to context properties

To directly observe how reference-sample selection affects prediction, we construct a two-dimensional, three-class Gaussian-mixture task with 256 fixed query points and 12 labeled candidate points per repeat. Selecting every possible context of 4, 5, or 6 candidates and predicting the same query batch with the official TabPFN-v3 predictor yields a complete prediction table of 6,633 contexts over three repeats, supporting matched comparisons of additions, exchanges, and paired additions. Even among the 216 size-six contexts containing exactly two rows per class, accuracy spans 59.8–98.0% in the first repeat with comparable spreads in the other repeats, so equal budgets and class counts leave substantial variation in context quality.

Local class support and its accumulation. For each added row, we inspect its label and distance to each query. Conditional on one same-class row already being present, closer same-class candidates yield larger average loss reductions in all three repeats (Figure 2a). Class presence itself is a prerequisite, as the predictor assigns negligible probability to omitted classes (Biloš et al., 2026; Lee, 2026). Holding the candidate and query fixed while changing the background, the candidate's gain shrinks once a same-class row is already present (Figure 2b), and 75.7–82.2% of class-complete comparisons matching the query class show a smaller loss reduction after the other row is added. Contributions thus depend on accumulated support, although both increasing and diminishing returns occur in the complete table.

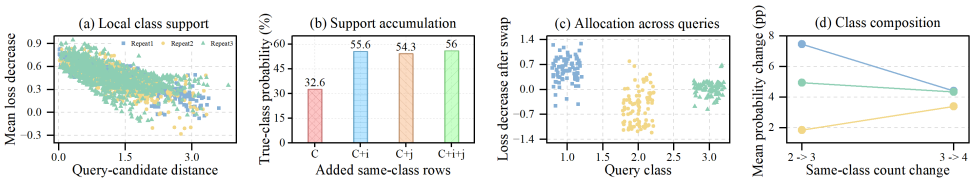

Figure 2: Context properties from exhaustive row comparisons. (a) Mean loss decrease versus query–candidate distance for same-class candidates over class-complete backgrounds with one row of the query's class. (b) A paired-addition example. (c) Query-level loss changes from a same-label swap retaining two rows per class, where positive values indicate improvement. (d) Mean true-class probability changes under background swaps with key local supports fixed. Panels (a,d) show all repeats and (b,c) show repeat-1 examples chosen by the rules in Appendix D.

Table 1: Factor responses (left) and selector mean log loss (right) on the five diagnostic tasks (TabPFN-v3, $B = 100$), averaged over six query batches per task. Near: loss reduction as the nearest same-class row moves from the within-class median rank to the nearest rank. P(0)/P(6): gain of a second-nearest same-class probe over a far control with 0 or 6 additional nearby supports. Alloc: equal regional allocation versus the mean of two extremes. Comp: true-class probability change (pp) as the same-class count rises from 2 to 16 (Appendix D). Bold marks the lowest loss per task.

	Factor responses					Selector mean log loss				
Task	Near	P(0)	P(6)	Alloc	Comp	Uniform	CB-Rand	Pooled kNN	Fac. Loc.	CRUMB
Helena	1.699	0.583	0.187	0.002	+18.68	9.307	**4.619**	9.910	9.100	13.475
Letter	2.194	0.077	0.004	0.410	+3.59	2.214	1.492	2.887	**1.119**	12.852
MNIST	1.313	0.240	0.011	0.262	+2.66	1.027	0.983	**0.743**	0.748	3.420
Volkert	0.981	0.487	0.143	0.195	+11.43	1.941	1.739	**1.651**	1.699	3.198
Connect-4	0.865	0.749	0.041	-0.009	-17.80	0.802	1.089	0.788	**0.780**	0.888

Allocation across queries and class composition. An exchange produces a vector of query-level gains. In the size-six same-label example of Figure 2c, 143 queries improve and 105 worsen by more than 0.02 loss, and over 99.7% of class-complete same-label swaps help at least ten queries and hurt at least ten, so evaluating an exchange requires considering how its gains and losses are distributed across queries. Comparing size-six contexts that preserve each query's two nearest same-class rows and nearest different-class row, raising the same-class count from two to three raises mean true-class probability by 1.84–7.46 percentage points across repeats (Figure 2d). Class composition accompanies prediction changes even with these local supports fixed.

These comparisons identify local class support, support accumulation, allocation across queries, and class composition as candidate properties for further analysis. Appendix D gives the complete setup, repeat statistics, and example-selection rules.

3.3 Checking the properties on real data

We examine the same properties on Helena, Letter, MNIST, Volkert, and Connect-4 using TabPFN-v3 and $B = 100$. Three frozen splits and two complete 256-query batches per split give 30 batches. Targeted comparisons cover 114 focal queries and 2,088 diagnostic contexts. All treatment definitions were frozen before these predictions were computed.

Moving the nearest same-class support closer reduces focal-query loss on all five tasks (Table 1, left), and the probe's gain decreases on every task as existing nearby support increases. These controls support the relevance of proximity and background-dependent accumulation beyond the toy setting.

Varying the allocation of context positions between two feature-defined query regions with class counts preserved, equal allocation improves whole-batch loss on Letter, MNIST, and Volkert, while effects on Helena and Connect-4 are small, with batch-bootstrap intervals spanning zero. Raising the total same-class count from 2 to 16 with the key local supports fixed increases mean true-class probability on four tasks and decreases it on Connect-4, with the MNIST interval spanning zero.

The value of balancing regional support and of composition shifts consequently depends on the task. Experimental details and additional results are provided in Appendix D.

3.4 Fixed factor weightings do not transfer across tasks

The preceding experiments identify local class support, support accumulation, allocation across queries, and class composition as properties that affect prediction, with response magnitudes that vary across tasks. Existing selectors build these factors into their rules to different extents. Comparing them under the shared-budget protocol (TabPFN-v3, $B = 100$, the same fixed query batches) shows that no rule wins throughout (Table 1, right). First place changes hands across tasks, and each hand-off follows the factor responses in the same rows of the table.

The winners are the rules whose built-in factors the task depends on. Helena spreads 100 classes over a budget of 100 rows, class coverage binds first, and Class-Balanced Random (CB-Rand), the only rule guaranteeing class support, halves the loss (4.62 versus 9.31). On MNIST and Volkert, every selector except CRUMB covers all query classes, so the advantage shifts to local support, and the two geometric selectors lead by placing more same-class rows near the queries. On Letter, where the regional-allocation controls find whole-batch gains from equal allocation, Facility Location (Lin & Bilmes, 2011; Wei et al., 2015) redirects the remaining budget toward the weakest queries and more than halves Pooled kNN's loss (1.12 versus 2.89).

The losses mirror the same matching. Every selector already covers the three classes of Connect-4, so the equal per-class allocation of Class-Balanced Random only moves the class composition away from the query batch, and it finishes last on the one task whose composition response is negative. Pooled kNN trails even Uniform Random on Helena (9.91 versus 9.31), and CRUMB, which matches the overall feature distribution of the query batch rather than any of the factors, is weakest on the remaining tasks. Appendix D reports the supporting context measurements.

These rules are therefore not competing heuristics but fixed points in a common design space. Each commits to one weighting of the factors, and no single weighting suits every task. The next section makes this space explicit: its configuration space recovers Class-Balanced Random, Pooled kNN, and Facility Location as corner settings (Appendix C) and interpolates between them, with the weighting selected by validation for each task.

4 Methodology

Given a labeled pool $\mathcal{D}$, a fixed query batch $\mathcal{Q} = \{q_i\}_{i=1}^{M}$, and a budget B, we select one context of B rows that the complete batch shares, and the frozen predictor predicts the batch in a single call. Section 3 identifies four factors that govern context quality, with response magnitudes that vary across tasks. We propose Preference-Aware Integrated Retrieval (PAIR), which pairs each factor with one adjustable mechanism, selected by validation for each task: the similarity function for local class support, the accumulation order p for support accumulation, the fairness exponent α for allocation across queries, and the reservation fraction f for class composition.

4.1 Preference-Aware Integrated Retrieval

Query coverage. The support accumulation of Section 3 means that the same candidate yields different gains in different contexts, so no score attached to a row in isolation can express its contribution. We therefore maintain one quantity per query that measures the evidence it receives from the selected set. Nonnegative similarities $k_{ij} = k(q_i, x_j) \geq 0$ are computed from query and candidate features, fixed before construction and shared by all classes, so selection needs no preliminary label prediction, and local class support enters the method through them. For the selected set S, define query coverage as

$$c_i(S) = \begin{cases} \left(\sum_{x_j \in S} k_{ij}^p \right)^{1/p}, & 1 \leq p < \infty, \\ \max_{x_j \in S} k_{ij}, & p = \infty, \end{cases} \tag{1}$$

with zero coverage for the empty set.

The accumulation order p controls how the similarities of several neighbors combine into coverage. At $p = 1$, they add directly. At $p = \infty$, the most similar selected neighbor determines coverage (Lin & Bilmes, 2011; Wei et al., 2015). Intermediate values let several neighbors contribute while adjusting the relative influence of stronger similarities. For $p > 1$, the same candidate yields different coverage increments in different contexts, matching the support accumulation observed in Section 3.

Coverage objective. The M coverages must combine into one objective. Section 3 shows that queries compete for the shared budget and that the value of balanced allocation varies across tasks, so the trade-off between queries is kept adjustable. At coverage level t, each additional unit of coverage receives weight $(\epsilon + t)^{-\alpha}$, with $\epsilon > 0$ and a finite fairness exponent $\alpha \geq 0$. The coverage objective is

$$F(S) = \sum_{i=1}^{M} \int_{0}^{c_i(S)} (\epsilon + t)^{-\alpha} \, \mathrm{d}t. \tag{2}$$

At $\alpha = 0$ every unit of coverage has equal weight and the objective sums the coverages. Positive α favors queries at lower coverage, so that F sums the α-fair utility of each query's coverage (Mo & Walrand, 2000; Rahmattalabi et al., 2021). Here *fair* denotes allocation of coverage across queries. We use $\epsilon = 10^{-3}$ to keep weights finite near zero coverage. The score increment of candidate x is

$$\begin{aligned} \Delta F(x \mid S) &= F(S \cup \{x\}) - F(S) \\ &= \sum_{i=1}^{M} \int_{c_i(S)}^{c_i(S \cup \{x\})} (\epsilon + t)^{-\alpha} \, \mathrm{d}t, \end{aligned} \tag{3}$$

which connects the two choices: the accumulation order p determines how much coverage the candidate adds, and the fairness exponent α determines the weight this coverage receives.

Class reservation. Coverage is a function of similarity alone, and the composition effects of Section 3 lie outside it: class composition shifts predictions even with local support held fixed, and classes absent from the context receive almost no probability. Query labels are unknown, so the context should preserve evidence from multiple classes. For a reservation fraction $f \in [0, 1]$, we allocate $b = \mathrm{round}(fB)$ positions as evenly as possible across classes, subject to available sample counts, and sample uniformly without replacement within each class to obtain the class-reserved set R. With $b \geq K$ and available rows, R covers all classes. The strength of these effects varies by task and f is adjusted accordingly. Rows in R also count toward coverage.

Greedy construction and guarantee. Starting from $S = R$, each iteration adds the unselected candidate with the largest score increment and updates query coverage until $|S| = B$.

Theorem 1 (Coverage objective and conditional greedy guarantee)**.** *For nonnegative similarities, $p \in [1, \infty]$, finite $\alpha \geq 0$, and $\epsilon > 0$, F is nonnegative, monotone, and submodular with $F(\varnothing) = 0$. Fix the class-reserved set R, let S_g be its greedy extension, and let $S_R^\star$ be a highest-scoring context containing R with size at most B. Then*

$$F(S_g) - F(R) \geq (1 - 1/e)\big[F(S_R^\star) - F(R)\big]. \tag{4}$$

Greedy extension from the class-reserved set thus obtains at least $1 - 1/e$ of the optimal score increment (Nemhauser et al., 1978), and diminishing gains enable lazy evaluation (Minoux, 1978). Appendices A and B give the proofs and the per-class quota guarantees.

4.2 Configuration selection and prediction

We jointly select the reservation fraction f, the accumulation order p, the fairness exponent α, and the similarity function on validation data. The experiments also evaluate the five supplementary cyclic rank-draft rules listed in Appendix C, which specifies the complete candidate set and validation procedure. The preceding guarantees apply to greedy construction under the coverage objective.

Class frequencies in the selected context can also affect model outputs (Zhao et al., 2021; Saerens et al., 2002; Lee, 2026). We provide an optional class-frequency correction and select its strength by

validation. Let $P_{\mathrm{TFM}}(c \mid q, S)$ be the model output and $h_S(c)$ the frequency of class c in the context. We consider $\beta \in \{0, 0.5, 1\}$. At $\beta = 0$, we retain the original output. For $\beta > 0$, represented classes receive

$$P_\beta(c \mid q, S) = \frac{P_{\mathrm{TFM}}(c \mid q, S) h_S(c)^{-\beta}}{\sum_{c': h_S(c')>0} P_{\mathrm{TFM}}(c' \mid q, S) h_S(c')^{-\beta}}, \qquad h_S(c) > 0. \tag{5}$$

Classes absent from the context receive zero mass under this correction.

The correction favors classes with lower context frequency, applies uniformly to the entire query batch, and requires no additional predictor calls, since all three β settings share one model output. Validation jointly selects context construction and correction strength before final testing. A selected configuration then uses one shared context and one prediction call per query batch.

For direct use with a fixed configuration, we provide the default $(f, p, \alpha) = (0.25, 2, 1)$, rank similarity, and $\beta = 0$.

5 Experiments

5.1 Experimental setup

We evaluate 51 prediction tasks from 50 datasets drawn from three broad tabular benchmark panels: TabArena-23, Multiclass-13, and OpenML Small-15 (Erickson et al., 2025; Vanschoren et al., 2014; Bischl et al., 2021). We retain classification tasks that satisfy common predictor and context-budget feasibility conditions, then freeze preprocessing, three outer splits, and non-overlapping query batches for every method. The primary budget is $B = 100$. Larger-budget results use the complete-case subset. Binary and multiclass tasks use ROC AUC and log loss, respectively, and scores are averaged over the three outer splits before per-task ranking (Demšar, 2006).

Every method returns one shared context per batch, and inner validation fixes all PAIR choices before final test scoring. The primary predictor is frozen one-estimator TabPFN-v3 (Grinsztajn et al., 2026), and the seven-method comparison includes PAIR, Untuned PAIR, Facility Location, Pooled kNN, Uniform Random, VIP-COp-lite-SR (Chen et al., 2026), and CRUMB ($K = 1$) (Heredge et al., 2026). Appendix C provides dataset, preprocessing, batch-size, and baseline details. Later experiments cover class-balanced ablations and three additional frozen predictors.

5.2 Main results

At $B = 100$, PAIR ranks first across 51 tasks with mean rank 1.91, followed by its untuned default at 2.61 and Facility Location at 3.65 (Table 2). The default's second place shows that the proposed retrieval objective performs competitively with a single configuration. Validation-selected PAIR leads the binary AUC and multiclass log-loss tasks with mean ranks 2.10 and 1.63, respectively. Excluding the five diagnostic-development tasks preserves the ordering (PAIR 1.98, untuned default 2.57, Facility Location 3.80), and Appendix G.1 reports two additional method sets and the interval calculation.

Gains beyond tuned simple selectors. We also compare PAIR with a matched-tuning comparator that chooses between Pooled kNN and Facility Location and jointly tunes the reservation fraction and the class-frequency correction, using the same predictor, budget, query batches, and validation data. Across the 31 binary tasks, mean AUC rises from 0.8082 to 0.8118, the mean gain magnitude is 2.33 times the mean loss magnitude, and omitting any single task preserves a positive mean gain. Across the 20 multiclass tasks, PAIR records 11 wins, five ties, and four losses, reducing mean log loss from 1.0058 to 0.9796, a relative reduction of 2.60%. Appendix G.5 reports per-task gains, confidence intervals, and sensitivity analyses.

5.3 End-to-End Inference Efficiency

Sharing one context per query batch reduces the time required to predict a complete test set. We compare PAIR, Uniform Random, Pooled kNN, Facility Location, CRUMB with 1, 4, and 16 clusters per batch, and per-query kNN on five datasets with frozen one-estimator TabPFN-v3 at $B = 100$,

Table 2: Mean rank [95% confidence interval] at $B = 100$ after averaging three splits for 51 tasks (31 binary AUC, 20 multiclass log loss), ranked among the seven methods with ties averaged, and lower is better. Untuned PAIR fixes $(f, p, \alpha) = (0.25, 2, 1)$, rank similarity, and $\beta = 0$.

Method	All 51	Binary 31	Multiclass 20
PAIR	**1.91** [1.63, 2.22]	**2.10** [1.71, 2.52]	**1.63** [1.30, 2.00]
Untuned PAIR	2.61 [2.27, 2.96]	2.65 [2.19, 3.10]	2.55 [2.05, 3.10]
Facility Location	3.65 [3.27, 4.02]	4.29 [3.84, 4.71]	2.65 [2.20, 3.10]
Pooled kNN	4.34 [3.87, 4.82]	4.68 [4.03, 5.29]	3.83 [3.20, 4.45]
Uniform Random	4.65 [4.27, 5.00]	4.42 [3.87, 4.97]	5.00 [4.70, 5.30]
VIP-COp-lite-SR	4.75 [4.12, 5.33]	3.94 [3.13, 4.74]	6.00 [5.50, 6.45]
CRUMB (K=1)	6.10 [5.71, 6.46]	5.94 [5.35, 6.45]	6.35 [5.85, 6.75]

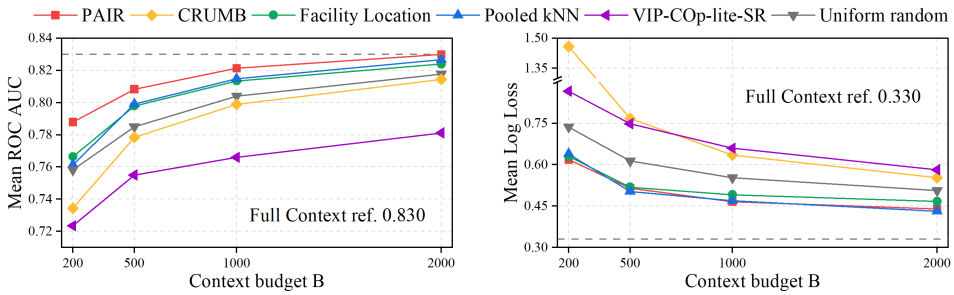

Figure 3: Official-metric performance versus context budget on 29 datasets (mean AUC over 17 binary tasks, log loss over 12 multiclass tasks), averaged over three splits and four frozen predictors. Dashed lines show Full Context on its feasible subset.

sharing test rows, hardware, and the parallel inference backend, averaging three splits per dataset, and weighting the five datasets equally (Figure 7 in Appendix H.1).

PAIR takes 48.7 seconds on average, against 66.8 and 140.5 seconds for the two clustered configurations and 1,223.9 seconds for per-query kNN, a $25.1\times$ speedup over per-query retrieval. Among the shared-context methods, which take 44.6–52.4 seconds, PAIR achieves the lowest mean log loss at 1.746, and per-query kNN reaches 1.393 at a much larger runtime.

5.4 Scaling with context budget

On the 29-dataset subset shared by four frozen predictors and all five budgets, PAIR ranks first at every evaluated budget, with mean ranks ranging from 1.70 at $B = 200$ to 2.63 at $B = 2000$. Its lead narrows as the context grows. Figure 3 shows the official-metric trends, and the complete rank table appears in Appendix G.7.

Exact Full Context provides an upper reference on the 19 feasible datasets, shown as dashed lines in Figure 3, and Appendix G.7 quantifies the remaining gap to the complete training pool.

PAIR stays first on the 36-task panel pooled over $B \in \{100, 200\}$, PAIR or its $\beta = 0$ variant ranks first on Small-15 at each $B \in \{64, 100, 128\}$ (Appendix G), and across the three large budgets PAIR combines the best mean within-budget rank with selection time comparable to Facility Location (Figure 8 in Appendix H).

5.5 Generality across predictors and task types

Across TabPFN-v3 (Grinsztajn et al., 2026), TabICL-v2 (Qu et al., 2026), TabFM (Kong & Das, 2026), and LimiX-16M (Zhang et al., 2025), the seven-method comparison including the $\beta = 0$ control gives PAIR the best mean ranks 1.93–2.46, followed by its $\beta = 0$ variant at 2.27–2.71, with Facility Location the strongest external baseline. TabPFN-v3 and TabICL-v2 cover all 51 tasks, TabFM and LimiX-16M cover 48, and complete per-predictor results appear in Appendix G.3.

We next evaluate whether coverage allocation extends beyond classification on eight frozen regression tasks with three splits, four predictors, and five budgets (100–2000). Regression isolates the coverage

Table 3: Component ablations at $B = 100$ over 51 tasks. Each variant restricts one parameter family and re-runs inner validation. W/T/L counts tasks where the full rule is better, equal, or worse on the official objective, and ties are tasks whose restricted grid re-selects the full configuration. The gap is the full rule's utility minus the variant's, and one-sided sign tests use non-tied tasks, uncorrected, with Bonferroni threshold 8.3×10^{-3} for six comparisons.

Restricted family	W/T/L	Median gap	Sign test
No class reservation ($f = 0$)	28/14/9	0.0040	1.3×10^{-3}
Reservation + Pooled kNN residual	37/2/12	0.0066	2.4×10^{-4}
Reservation + Facility Location residual	32/2/17	0.0034	2.2×10^{-2}
No fairness weighting ($\alpha = 0$)	17/22/12	0.0000	2.3×10^{-1}
Fixed accumulation order ($p = 1$)	28/13/10	0.0032	2.6×10^{-3}
No correction ($\beta = 0$, re-selected)	23/22/6	0.0000	1.2×10^{-3}

component: PAIR uses validation-selected p, α, and similarity function, without the classification-specific class reservation and class-frequency correction, against Uniform Random, Pooled kNN, Facility Location, and CRUMB ($K = 1$). Ranking methods by RMSE within each dataset–predictor–budget cell, PAIR ranks first at every budget and under every predictor, with an overall mean rank of 1.70 against 2.23 for Facility Location, the strongest baseline (Table 15 in Appendix G.4).

5.6 Ablations of selection freedoms

Section 3 motivates three adjustable choices: how coverage accumulates over neighbors, how coverage is allocated across queries, and how much of the budget is reserved across classes. We restrict each corresponding parameter family in turn to assess the predictive value of retaining these choices, re-run inner validation over its allowed configurations, and evaluate on the same test tasks. Table 3 reports paired outcomes over 51 tasks at $B = 100$.

Four of the six restrictions are worse than the full rule at the Bonferroni-adjusted threshold: removing class reservation ($f = 0$), replacing greedy coverage selection with a Pooled kNN residual, fixing the accumulation order ($p = 1$), and disabling the class-frequency correction with the retrieval configuration re-selected. The Facility Location residual is directionally worse but does not pass the adjusted threshold. These outcomes support adjustable class reservation, flexible residual selection, and alternative accumulation orders, in accord with the observations of Section 3.

Aggregate evidence for fairness weighting is weaker: fixing $\alpha = 0$ gives 17 wins, 22 ties, and 12 losses, short of significance, yet with the other parameters fixed $\alpha = 0.5$ reduces mean log loss on each of the five development tasks (Figure 6 in Appendix G.10), so we retain validation-selected α. Overall, class reservation and coverage accumulation carry the clearest cross-task evidence, the optional class-frequency correction benefits some tasks at no additional model call, and Appendix G.9 reports mean ranks and measurement notes for every restricted family.

6 Limitations and Conclusion

We study real-row retrieval for frozen TFMs with fixed query batches and standardized Euclidean distances, and the diagnostics cover one synthetic family and five development tasks. Geometric coverage is a modeling choice evaluated empirically, the approximation guarantee is conditional on the realized class-reserved set, and configuration selection requires inner validation, with the class-frequency correction varying in effectiveness on small data. Reducing selection cost and studying distribution shifts, learned distances, and more flexible query batches are future directions.

Our analyses relate sample contributions to local class support, support accumulation, allocation across queries, and class composition, with task-dependent responses that fixed selection rules cannot follow. PAIR organizes these factors into an adjustable framework chosen on validation data, and the prediction, efficiency, and ablation results across tasks, budgets, and predictors support the practical value of shared-context retrieval.

REFERENCES

Marin Biloš, James T Wilson, Anderson Schneider, and Yuriy Nevmyvaka. A mechanistic study of tabular foundation models. *arXiv preprint arXiv:2605.21288*, 2026.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Yilong Chen, Xueying Ding, and Leman Akoglu. VIP-COP: Context optimization for tabular foundation models. *arXiv preprint arXiv:2605.12904*, 2026.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

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

Benjamin Feuer, Robin T Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. TuneTables: Context optimization for scalable prior-data fitted networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28, 2015.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13:723–773, 2012.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. TabPFN-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Jamie Heredge, Mattia J Villani, Pranav Deshpande, Akshay Seshadri, and Niraj Kumar. CRUMB: Efficient prior fitted network inference via distributionally matched context batching. *arXiv preprint arXiv:2606.11473*, 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*, 2023.

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

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. *Google Research Blog*, 30, 2026.

Andreas Krause and Daniel Golovin. Submodular function maximization. In Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli (eds.), *Tractability: Practical Approaches to Hard Problems*, pp. 71–104. Cambridge University Press, Cambridge, 2014.

Seunghan Lee. Mitigating label shift in tabular in-context learning via test-time posterior adjustment. *arXiv preprint arXiv:2605.04363*, 2026.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 510–520, 2011.

Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pp. 3122–3130. PMLR, 2018.

Majed Luay, Siamak Layeghy, Seyedehfaezeh Hosseininoorbin, Mohanad Sarhan, Nour Moustafa, and Marius Portmann. Temporal analysis of NetFlow datasets for network intrusion detection systems. *arXiv preprint arXiv:2503.04404*, 2025.

Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with TabPFN. *arXiv preprint arXiv:2402.06971*, 2024.

Ratmir Miftachov, Bruno Charron, and Simon Valentin. Interpretable tabular foundation models via in-context kernel regression. *arXiv preprint arXiv:2602.02162*, 2026.

Michel Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization Techniques*, volume 7 of *Lecture Notes in Control and Information Sciences*, pp. 234–243. Springer, 1978.

Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on networking*, 8(5):556–567, 2000.

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

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical programming*, 14(1):265–294, 1978.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 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.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

Aida Rahmattalabi, Shahin Jabbari, Himabindu Lakkaraju, Phebe Vayanos, Max Izenberg, Ryan Brown, Eric Rice, and Milind Tambe. Fair influence maximization: A welfare optimization approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11630–11638, 2021.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ digital medicine*, 1(1):18, 2018.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

Nour Shaheen, Junwei Ma, Alex Labach, Frank Hutter, Valentin Thomas, and Anthony L Caterini. Understanding the surprising generalization properties of tabular foundation models. *arXiv preprint arXiv:2608.17957*, 2026.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony Caterini. Retrieval & fine-tuning for in-context tabular models. *Advances in Neural Information Processing Systems*, 37:108439–108467, 2024.

Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pp. 1954–1963. PMLR, 2015.

Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & manufacturing research*, 4 (1):23–45, 2016.

Derek Xu, Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. In *International Conference on Learning Representations*, volume 2025, pp. 24630–24665, 2025.

Yuchen Zeng, Tuan Dinh, Wonjun Kang, and Andreas C Mueller. TabFlex: Scaling tabular learning to millions with linear attention. *arXiv preprint arXiv:2506.05584*, 2025.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, et al. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pp. 12697–12706. Pmlr, 2021.

A Monotonicity and Submodularity of the Coverage Objective

This appendix proves the objective properties stated in Theorem 1. The integral in Equation 2 has the closed form

$$u_\alpha(z) = \begin{cases} \log\dfrac{\epsilon + z}{\epsilon}, & \alpha = 1, \\ \dfrac{(\epsilon + z)^{1-\alpha} - \epsilon^{1-\alpha}}{1-\alpha}, & \alpha \neq 1, \end{cases} \tag{6}$$

with $F(S) = \sum_i u_\alpha(c_i(S))$. The fundamental theorem of calculus gives

$$\int_0^z (\epsilon + t)^{-\alpha}\,\mathrm{d}t = u_\alpha(z), \qquad u'_\alpha(z) = (\epsilon + z)^{-\alpha}, \qquad u''_\alpha(z) = -\alpha(\epsilon + z)^{-\alpha-1}.$$

Thus the integral and closed form define the same normalized objective. For finite $\alpha \geq 0$, u_α is nonnegative, nondecreasing, and concave on the nonnegative reals, with $u_\alpha(0) = 0$. In particular, each candidate increment equals

$$F(S \cup \{x\}) - F(S) = \sum_i \int_{c_i(S)}^{c_i(S\cup\{x\})} (\epsilon + t)^{-\alpha}\,\mathrm{d}t.$$

We prove submodularity query by query. Fix i and first let $1 \leq p < \infty$. Write $a_j = k_{ij}^p \geq 0$ and

$$g(t) = u_\alpha(t^{1/p}), \qquad t \geq 0.$$

The map $t \mapsto t^{1/p}$ is concave and nondecreasing for $p \geq 1$. Its composition with the concave nondecreasing u_α is concave and nondecreasing, with $g(0) = 0$ (Krause & Golovin, 2014). Hence

$$G_i(S) = g\left(\sum_{j\in S} a_j\right)$$

has diminishing returns: for $A \subseteq C$ and $x \notin C$, the nonnegative increment a_x and concavity give

$$G_i(A \cup \{x\}) - G_i(A) \geq G_i(C \cup \{x\}) - G_i(C).$$

Nonnegativity, normalization, and monotonicity follow from the corresponding properties of g.

For $p = \infty$, monotonicity of u_α gives

$$G_i(S) = u_\alpha\left(\max_{j\in S} k_{ij}\right) = \max_{j\in S} u_\alpha(k_{ij}),$$

with zero for the empty set. A nonnegative weighted maximum is monotone submodular. Finally, $F = \sum_i G_i$, and sums preserve all four properties. The $1 - 1/e$ cardinality-constrained greedy guarantee follows from Nemhauser et al. (1978). The five cyclic rank-draft rules in the validation grid are direct heuristics. This theorem applies to the finite-α coverage family.

B Conditional Greedy Guarantee of Theorem 1 and Per-Class Quotas

B.1 Balanced class reservation

For any feasible per-class quota vector, transferring a position from a larger component to a smaller, uncapped component that differs by at least two improves the nondecreasing ordering lexicographically. Repeated transfers produce balanced quotas. Progressive filling handles class capacities by assigning remaining positions among eligible classes. This allocation maximizes the number of represented classes. A uniform reference label is unsupported with probability equal to the fraction of zero quota components, which the balanced allocation minimizes. Every class is represented when $b \geq K$ and each class has at least one available row.

B.2 Conditional greedy guarantee

Fix a realized class-reserved set R. The conditional selection problem is

$$\max_{\substack{R\subseteq S\subseteq\mathcal{D}\\ |S|\le B}} F(S). \tag{7}$$

Define, on $\mathcal{D}\setminus R$,

$$H(T)=F(R\cup T)-F(R).$$

Then $H(\varnothing)=0$, and monotonicity of F makes H nonnegative and monotone. For $A\subseteq C\subseteq\mathcal{D}\setminus R$ and $x\notin R\cup C$, submodularity of F gives

$$H(A\cup\{x\})-H(A)\ge H(C\cup\{x\})-H(C).$$

Thus H is submodular. Let $T^\star$ maximize H under $|T|\le B-|R|$, and let T_g be the greedy extension. The cardinality-constrained guarantee yields

$$F(R\cup T_g)-F(R)\ge(1-1/e)\big[F(R\cup T^\star)-F(R)\big].$$

For zero remaining budget, both increments are zero. The inequality holds for every realized R. Taking expectations preserves it, with the optimal extension defined for the same realized class-reserved set.

B.3 A class-compatible interpretation of query coverage

For a possible query label c, define

$$e_{ic}(S)=\Big\|\big(k_{ij}\big)_{j\in S:y_j=c}\Big\|_p, \tag{8}$$

with zero evidence for an absent class. The true query label is unavailable during selection. For finite p, class-specific subsets partition S, so a uniform reference label C_i gives the identity

$$K\,\mathbb{E}[e_{iC_i}(S)^p]=\sum_{c=1}^{K}e_{ic}(S)^p=\sum_{j\in S}k_{ij}^p=c_i(S)^p.$$

For $p=\infty$, maximizing within classes and then across classes gives $c_i(S)=\max_c e_{ic}(S)$. This supplies a class-compatible interpretation of the geometric coverage in Equation 1. The resulting coverage depends on similarities. Explicit class allocation enters through R in Equation 7.

C Complete Experimental Protocol

C.1 Data, splits, and fixed query batches

The primary federation contains 51 prediction tasks from 50 raw datasets. The TabArena panel contributes 23 tasks, the independently assembled multiclass panel contributes 13, and OpenML Small-15 contributes 15. Two prediction tasks use different target columns from one source table and are counted separately because they define different supervised problems. The combined set contains 31 binary and 20 multiclass objectives.

Each task uses three outer splits frozen before the full-grid runs. For TabArena-23 and Multiclass-13, the validation set contains at most 1,024 rows, the test set at most 2,048, and a fixed query batch at most 256. Small-15 uses caps of 256, 512, and 16, respectively, and retains at least two nonempty query batches. Query order and grouping are identical for every selector. One exact context is stored for each batch, and the complete batch is passed to the predictor in one call. Contexts contain B unique row identifiers from the outer training split. The primary budget is $B=100$, the only budget shared by all three panels. The 36 TabArena-23 and Multiclass-13 tasks also use $B=200$, and Small-15 uses $B\in\{64,100,128\}$. The budget-scaling subset contains 29 datasets (19 from TabArena-23 and 10 from Multiclass-13) evaluated with four predictors at $B\in\{100,200,500,1000,2000\}$; the remaining TabArena-23 tasks have training pools too small for $B=2000$. The 8-task regression panel uses the same five budgets.

C.2 PREPROCESSING AND PREDICTORS

Selector preprocessing is fitted on training features only. Numeric features are standardized with training means and variances, after which an exact Euclidean nearest-neighbor cache of depth 256 (128 for Small-15) is constructed for validation and test queries. The same cache and deterministic row-id tie break are reused by Pooled kNN, Facility Location, the cyclic rank-draft rules, and PAIR. Pool labels are encoded from the training split. Missing-value and categorical preprocessing follow the frozen benchmark manifests and are shared by every method.

The primary predictor is the frozen `tabpfn-v3-classifier-v3_default` checkpoint (Grinsztajn et al., 2026) from package version `8.0.8`, run with one estimator. Every method changes only the context row ids. The predictor comparison adds the frozen `tabicl-classifier-v2-20260212` checkpoint (Qu et al., 2026) and the TabFM 1.0.0 classification weights (Kong & Das, 2026), each run with one estimator, and uses the frozen `LimiX-16M.ckpt` checkpoint (Zhang et al., 2025) and its no-retrieval classification configuration. TabFM and LimiX-16M support at most ten classes, so they omit Helena, Devanagari Script, and Letter and cover 48 of the 51 tasks. Five of 276 LimiX run shards encountered an ARPACK failure in preprocessing. The registered fallback changed the affected decomposition to the randomized solver. This covered 32 query-batch predictions. The fallback count and batch ids were persisted before aggregation.

C.3 BASELINE IMPLEMENTATIONS

Pooled kNN assigns the row at rank r, zero-indexed, weight $1/(r+1)$ for each query and selects the B rows with the largest summed weight. Facility Location uses $k(q,x) = \exp(-d(q,x)/s)$, with s equal to the median distance to the 16th cached neighbor, and applies lazy greedy selection (Minoux, 1978). Class-Balanced Random distributes B positions evenly across training classes and samples without replacement with a batch-specific frozen seed. Small-15 uses capacity-capped equal quotas. The other panels use equal quotas with a deterministic remainder. CRUMB uses one complete frozen query batch as its single cluster, 64 random Fourier features (Rahimi & Recht, 2007), greedy batches of 50, and 20,000 pairs for bandwidth estimation (Gretton et al., 2012), matching the registered $K = 1$ implementation of the published selector.

VIP-COp-lite-SR runs the released row-selection core of Chen et al. (2026) (Gumbel top-k coalition sampling, centered validation rewards, cumulative diagonal ridge estimation, and confidence-bound pruning) for 30 rounds with 64 coalitions of size 50 per round, giving $30 \times 64 = 1{,}920$ logical coalition evaluations per dataset and split. The reward is the mean true-class margin on a validation cap of 256 queries. The resulting row ranking is shared across budgets, with smaller budgets taken as nested prefixes of the largest-budget ranking. This is the "SR" (shared ranking) protocol. The implementation is release-inspired rather than a full reproduction: the released default of 2,000 coalitions per round for 400 rounds is reduced to keep the baseline feasible across 51 tasks and four predictors. Because of this reduction, VIP-COp-lite-SR is treated as a reference reading rather than as a primary baseline, and no headline conclusion depends on the size of its gap to PAIR.

C.4 THE PAIR GRID AND VALIDATION

The 78 context selectors are composed as follows:

- 72 finite-α coverage settings from $f \in \{0, .25, .5, .75\}$, $p \in \{1, 2, \infty\}$, $\alpha \in \{0, 1, 2\}$, and rank or exponential similarity,
- five $f = 0$, $p = 1$, cyclic rank-draft rules with representative ratios $.25, .5, .8, 1, 3$, and
- the pure $f = 1$ class reservation.

Both similarities are evaluated on each query's cached neighbor list and are zero for rows outside it, so candidates are drawn from the union of the cached neighbors. Rank similarity assigns the neighbor at zero-indexed rank r the value $k_{ij} = 1/(r+1)$, and exponential similarity uses $k_{ij} = \exp(-d(q_i, x_j)/s)$ with the Facility Location scale s above. The search space recovers Class-Balanced Random at $f = 1$, Pooled kNN at $(f, p, \alpha) = (0, 1, 0)$ with rank similarity, and Facility Location at $(0, \infty, 0)$ with exponential similarity. We use $\epsilon = 10^{-3}$ in Equation 6. Each cyclic

rank-draft rule starts from a deterministic farthest-first traversal (Gonzalez, 1985) of standardized query features, retains $\min(M, \mathrm{round}(rB))$ representatives for its registered ratio r, and cycles through their ranked neighbors until the context is full. Finite candidates optimize over every query in the fixed batch. Each selected context yields three prediction rules with $\beta \in \{0, .5, 1\}$, for 234 validation candidates but only 78 distinct contexts and TFM calls. Candidate scores are averaged across three inner validation splits and optimized in the direction of the task's official metric. Tie breaks are deterministic and fixed before outer testing. The uncorrected ablation repeats selection with $\beta = 0$ and uses the same context grid.

For the optional class-frequency correction, we retain the original predictor probabilities at $\beta = 0$. At $\beta \in \{0.5, 1\}$, Equation 5 reweights represented classes by their empirical context frequencies. Absent classes receive zero mass. This output-level correction follows the label-shift adjustment line (Saerens et al., 2002; Lipton et al., 2018; Shimodaira, 2000). Lee (2026) recently applied test-time posterior adjustment to tabular in-context learning. Our correction uses only the selected context's empirical class frequencies and is chosen by inner validation. It shares the equal-label reference of class reservation and acts only on model outputs. The submodular selection guarantee applies to the coverage objective F.

The complete protocol asserts that validation and test row ids are disjoint from the pool, every selected context has the exact requested size, class reservations are satisfied, and all methods use identical query batches. Test labels are read only by the metric writer after predictions and selected-candidate records have been persisted.

D Context Properties: Exhaustive and Real-Data Diagnostics

D.1 Exhaustive synthetic protocol

Three two-dimensional classes have centers $(-1.5, -0.8)$, $(1.5, -0.8)$, and $(0, 1.6)$. Each class contains two isotropic Gaussian components, positioned at offsets of ± 0.7 along a random direction from its center, with component standard deviation 0.5. Two training rows per component yield 12 candidates. Each repeat independently generates 256 queries with class counts $(86, 85, 85)$ and equally probable within-class components. Generation seeds are 1701, 1702, and 1703. All generated repeats are retained. Figure 4 displays their complete inputs.

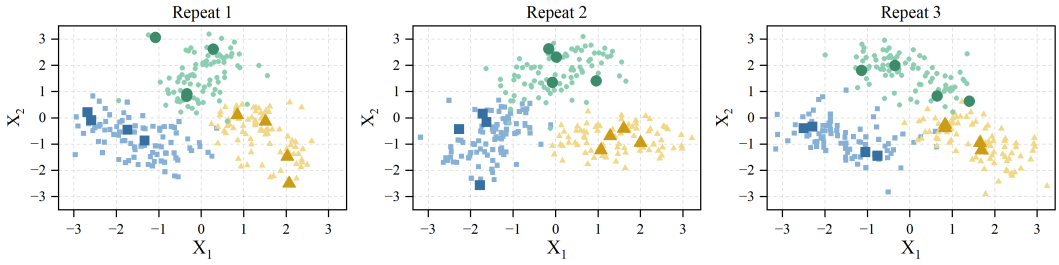

Figure 4: Complete synthetic inputs. Colors denote classes, small translucent points are queries, and outlined points are the 12 training candidates.

For each repeat, we evaluate all $\binom{12}{4} = 495$, $\binom{12}{5} = 792$, and $\binom{12}{6} = 924$ contexts, totaling 2,211 contexts and 566,016 query predictions. Three repeats give 6,633 contexts and 1,698,048 predictions. The 256 queries remain in one fixed batch with fixed row order. Context rows are sorted by original row identifier. The model uses TabPFN-v3, checkpoint `tabpfn-v3-classifier-v3_default`, package version 8.0.8, one estimator, and official preprocessing and precision. Inference seeds 101, 102, and 103 are constant across contexts within their respective repeats. Saved predictions match the fixed serial-reference contexts exactly. All probability arrays pass shape, finite-value, row-order, and normalization checks. The maximum row-sum error is 4.47×10^{-8}.

For query q, let $\ell_q(C) = -\log P(y_q \mid q, C)$. We define an addition gain as $\ell_q(C) - \ell_q(C \cup \{i\})$, and an exchange gain analogously. The paired-addition contrast is

$$I_q(C; i, j) = \ell_q(C) - \ell_q(C \cup \{i\}) - \ell_q(C \cup \{j\}) + \ell_q(C \cup \{i, j\}).$$

Positive I_q means that adding i has a smaller loss benefit after j is present. We report the fraction exceeding 10^{-8}. This is a conditional empirical comparison for the frozen predictor. For example, across all class-complete backgrounds and unrestricted candidate labels, the positive fraction is substantially lower than for pairs matching the query class.

Table 4: Complete repeat summaries. Accuracy spans all 216 size-six contexts with two rows per class. Diminishing gain conditions on a class-complete size-four background and two added rows matching the query class. Mixed query effects cover all class-complete, same-label size-six swaps, requiring at least ten gains above 0.02 and ten below -0.02.

Repeat	Accuracy range (%)	Diminishing gain (%)	Mixed query effects (%)
1	59.8–98.0	77.5	99.81
2	78.1–97.3	82.2	99.79
3	65.6–96.9	75.7	99.93

Illustrative example rules. Figure 2b uses repeat 1 and the positive same-query-class interaction closest to the 90th percentile among 133,368 positive comparisons on class-complete backgrounds. Enumeration order breaks ties. Its background row indices are $\{0, 5, 7, 9\}$, added rows are 1 and 3, and query index is 132. The four true-class probabilities are 0.326300, 0.555658, 0.543055, and 0.560420. The corresponding loss gains for adding row 1 are 0.532337 and 0.031475 before and after row 3 is present. Figure 2c uses the same repeat and exchanges row 4 for row 6 in $\{2, 3, 4, 7, 9, 10\}$. Among 606 same-label swaps helping and hurting at least ten queries by 0.02 loss and canceling at least 90% of absolute gain, this example maximizes mean absolute query gain. Ties follow enumeration order. Cancellation is $1 - |\operatorname{mean}(g)|/\operatorname{mean}(|g|)$, where g is the query-gain vector. The example cancels 93.22% of absolute gain. These examples illustrate substantial conditional and query-level changes. The complete repeat summaries establish their wider frequency within this toy.

Composition comparisons. We enumerate size-six context exchanges retaining the identities of the query's two nearest same-class training rows and nearest different-class training row. Comparisons are grouped by same-class count changes $2 \to 3$ and $3 \to 4$. Figure 2d averages true-class probability changes over all eligible query–exchange pairs in each repeat. The compared distant rows carry both labels and features. This protocol measures their combined contribution with the named local supports fixed. The synthetic analysis covers one distribution family and small budgets. Its findings motivate the real-data controls below.

D.2 Real-data controls and aggregation

The exploratory diagnostic uses Helena, Letter, MNIST, Volkert, and Connect-4, five tasks that participated in method development, with three frozen splits and two complete validation batches per split. Each batch contains 256 queries and is evaluated jointly with its original row order. Each final context contains 100 distinct training rows. Within each batch, we uniformly sample up to four classes having at least 32 training rows and one focal query per sampled class, giving 114 focal units overall. Query labels define the diagnostic treatments. Existing retrieval methods retain their original information access during selection.

All treatments for a batch use the same model seed, official TabPFN-v3 checkpoint, preprocessing, precision, and single estimator. The controls produce 2,088 diagnostic contexts. Method profiling evaluates 180 contexts including a supplementary PAIR comparison. Reference calls are additional to these counts. The largest deviation from cached reference probabilities is 2.11×10^{-8}, and saved probability row-sum errors are below 5.4×10^{-8}. Data, splits, and query batches are unchanged.

Local class support. With 99 shared rows outside the focal class, the final slot contains either another different-class row, a far same-class row, or the nearest same-class row. A second control fixes eight same-class slots and 92 other rows. The nearest selected same-class row is placed at within-class distance percentiles 0, 10, or 50. Seven common same-class rows come from the farthest quarter. The main table reports the loss reduction from percentile 50 to percentile 0. The omitted-class response includes the predictor's probability floor.

Support accumulation. We fix the nearest same-class row, reserve the second-nearest as a probe, and retain eight same-class slots. Six other slots contain 0, 1, 3, or 6 nearby same-class rows, with far rows filling the remainder. At each level, a matched pair exchanges a far same-class control for the probe. The gain is the resulting focal-query loss reduction. Both class count and context size are identical within and across levels.

Allocation across queries. The median of the highest-variance feature partitions each complete query batch into two fixed regions. At fixed class counts, fractions 0, 0.25, 0.5, 0.75, and 1 of the context positions are assigned to region A, with the remainder assigned to region B. Within each class, candidates are ordered by distance to the nearest query in the designated region. Selection preserves unique rows. The allocation contrast is the mean whole-batch loss of the two extreme allocations minus the loss at equal allocation.

Class composition. We retain the two nearest same-class rows and nearest different-class row, and exchange distant different-class rows for distant same-class rows to obtain focal-class counts 2, 8, and 16. Every donating class retains at least one row. The realized controls preserve the same-class count among the nearest 32 full-pool neighbors in all 114 focal units. The probability contrast is $P(y_q \mid C_{16}) - P(y_q \mid C_2)$. Other background features also change.

Aggregation and uncertainty. We first average focal-unit effects within each complete query batch, then weight the six batches equally within each task. Tables below show percentile 95% intervals from 20,000 bootstrap resamples of these six batch effects. These intervals describe the sampled batches. Batches across splits can overlap, and six batches provide limited information about population-level uncertainty. All five tasks and all defined contrasts are retained in the source results.

Table 5: Real-data control means and descriptive 95% batch-bootstrap intervals. Column definitions match the left block of Table 1. Comp is in percentage points and other columns are loss reductions.

Task	Near	P(0)	P(6)	Alloc	Comp
Helena	1.699 [1.076, 2.245]	0.583 [0.406, 0.734]	0.187 [0.155, 0.222]	0.002 [-0.012, 0.017]	18.677 [11.817, 24.936]
Letter	2.194 [1.869, 2.592]	0.077 [0.051, 0.103]	0.004 [0.001, 0.008]	0.410 [0.327, 0.494]	3.593 [1.691, 5.587]
MNIST	1.313 [0.779, 1.750]	0.240 [0.123, 0.353]	0.011 [-0.007, 0.029]	0.262 [0.183, 0.351]	2.662 [-5.358, 11.583]
Volkert	0.981 [0.627, 1.324]	0.487 [0.326, 0.648]	0.143 [0.092, 0.212]	0.195 [0.070, 0.330]	11.428 [5.485, 20.737]
Connect-4	0.865 [0.725, 1.020]	0.749 [0.601, 0.914]	0.041 [0.029, 0.049]	-0.009 [-0.087, 0.069]	-17.804 [-26.898, -10.609]

D.3 EXISTING-METHOD PROFILES

For each method we measure true-class absence, the nearest same-class row's full-pool distance percentile conditional on class presence, and the selected same-class count among the nearest 32 full-pool neighbors. Define the nearest-support rank as the reciprocal full-pool rank of its nearest selected same-class row, assigning zero when that class is absent. Weak support is the mean of this quantity for the method's own weakest quarter of queries. Context–query frequency distance is $\frac{1}{2}\sum_c |h_C(c) - h_Q(c)|$. All quantities are computed within a batch and then averaged equally over six batches per task. The weakest quartiles and class-present subsets can differ across methods. Table 6 gives all five baseline profiles, including loss. The profile variables describe context properties without combining them into a single predictor of method rankings.

These profiles accord with the per-task loss differences in Table 1. On Helena, Uniform Random's true-class absence accompanies its high loss, and the composition controls suggest that additional class reservation should account for the coverage the context already provides, as on Connect-4. On MNIST, the geometric selectors' smaller nearest same-class distances accompany their lower loss with class coverage complete. On Letter, Facility Location provides stronger support for its weakest quarter of queries than Pooled kNN, consistent with the whole-batch gains of equal regional allocation.

Table 6: Existing-method profiles on all diagnostic tasks. Missing: queries without a same-class row (%); rank: nearest same-class full-pool percentile, conditional on presence; near32: selected same-class count among the nearest 32 rows; weak: weakest-quartile reciprocal-rank support, multiplied by 10^3; TV: context–query class-frequency distance.

Task	Method	Loss	Missing	Rank	Near32	Weak	TV
Helena	CRUMB (K=1)	13.475	41.992	31.354	0.018	0.000	0.562
	CB-Rand	4.619	0.000	33.805	0.008	0.024	0.441
	Fac. Loc.	9.100	24.609	15.396	0.127	0.002	0.348
	Pooled kNN	9.910	28.125	14.234	0.141	0.000	0.381
	Uniform	9.307	25.065	17.718	0.013	0.002	0.372
Letter	CRUMB (K=1)	12.852	41.341	26.542	0.161	0.000	0.613
	CB-Rand	1.492	0.000	7.477	0.158	0.327	0.112
	Fac. Loc.	1.119	0.000	5.550	0.558	0.554	0.142
	Pooled kNN	2.887	6.250	6.821	0.711	0.219	0.218
	Uniform	2.214	2.214	10.029	0.124	0.237	0.218
MNIST	CRUMB (K=1)	3.420	7.161	44.343	0.027	0.012	0.343
	CB-Rand	0.983	0.000	3.100	0.045	0.298	0.074
	Fac. Loc.	0.748	0.000	1.330	0.413	2.081	0.113
	Pooled kNN	0.743	0.000	2.357	0.532	0.450	0.133
	Uniform	1.027	0.000	3.231	0.046	0.272	0.154
Volkert	CRUMB (K=1)	3.198	5.599	18.047	0.033	0.034	0.252
	CB-Rand	1.739	0.000	5.448	0.027	0.168	0.318
	Fac. Loc.	1.699	0.521	5.762	0.336	0.170	0.196
	Pooled kNN	1.651	0.260	4.092	0.339	0.266	0.132
	Uniform	1.941	1.172	6.471	0.035	0.148	0.135
Connect-4	CRUMB (K=1)	0.888	0.000	8.240	0.021	0.100	0.050
	CB-Rand	1.089	0.000	2.762	0.019	0.295	0.346
	Fac. Loc.	0.780	0.000	1.637	0.329	0.487	0.045
	Pooled kNN	0.788	0.000	2.184	0.369	0.351	0.037
	Uniform	0.802	0.000	2.345	0.034	0.343	0.040

Table 7: Probe gains at all four existing-support levels, with descriptive 95% batch-bootstrap intervals. Eight same-class slots are retained.

Task	0 nearby	1 nearby	3 nearby	6 nearby
Helena	0.583 [0.406, 0.734]	0.399 [0.291, 0.497]	0.192 [0.128, 0.252]	0.187 [0.155, 0.222]
Letter	0.077 [0.051, 0.103]	0.027 [0.015, 0.046]	0.023 [0.004, 0.057]	0.004 [0.001, 0.008]
MNIST	0.240 [0.123, 0.353]	0.121 [0.052, 0.196]	0.069 [0.017, 0.156]	0.011 [-0.007, 0.029]
Volkert	0.487 [0.326, 0.648]	0.309 [0.232, 0.401]	0.240 [0.164, 0.336]	0.143 [0.092, 0.212]
Connect-4	0.749 [0.601, 0.914]	0.625 [0.393, 0.866]	0.223 [0.166, 0.281]	0.041 [0.029, 0.049]

E Selection Geometry and Illustrative Ranking Reversals

To expose how a shared context budget is allocated, we compare four selectors under the same one-context constraint. *Uniform Random* ignores query geometry. *Pooled kNN* rewards rows that are close to many queries. *Facility Location* rewards coverage of queries not yet well represented (Nemhauser et al., 1978; Wei et al., 2015). *CRUMB ($K = 1$)* matches the feature distribution of the complete batch (Heredge et al., 2026). This label follows CRUMB's own cluster-count notation, whereas elsewhere K is the number of classes. Each method returns B distinct real rows for one

frozen-model call. Figure 5 visualizes their contrasting spatial preferences, and exact implementations are given in Appendix C.

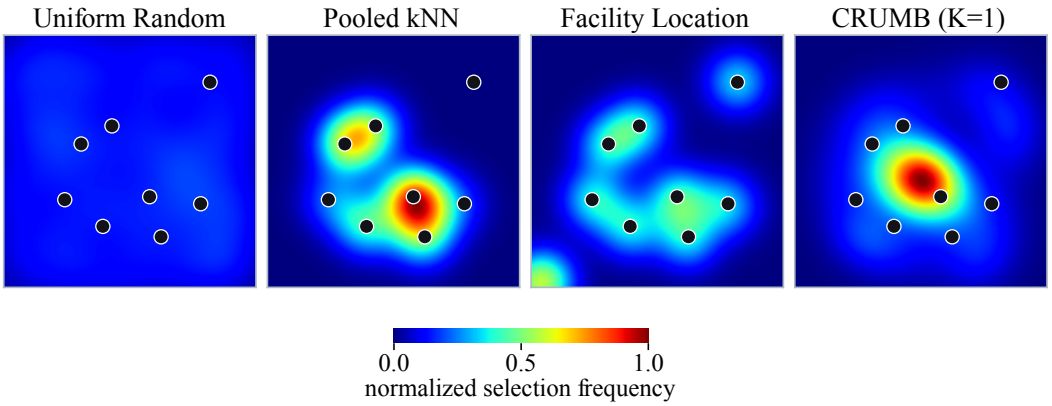

Figure 5: Selection geometry under a shared context budget. Black dots show the same query batch, and each method selects $B = 10$ rows from a common uniform candidate pool. Heatmaps report Gaussian-smoothed selection frequencies over 180 perturbed trials on a shared scale. Uniform Random remains diffuse, Pooled kNN concentrates on query-dense regions, Facility Location spreads coverage across queries, and CRUMB ($K = 1$) matches the batch distribution.

E.1 Illustrative ranking reversals

Five illustrative tasks from the full benchmark reveal clear reversals among the three query-conditioned selectors (Table 8). Facility Location leads on Credit Card Default and Segment, Pooled kNN on Fashion-MNIST and Airline Satisfaction, and CRUMB on Australian. These tasks serve as illustrative examples within the complete 51-task evaluation, with method design, hyperparameters, and evaluation settings fixed in advance under a common protocol.

Table 8: Within-task ranks of three query-conditioned selectors on five illustrative tasks at $B = 100$, after averaging the official metric over three frozen splits. Lower is better.

Dataset	Metric	Pooled kNN	Facility Location	CRUMB ($K = 1$)
Credit Card Default	ROC AUC	2	**1**	3
Fashion-MNIST	Log loss	**1**	2	3
Segment	Log loss	2	**1**	3
Airline Satisfaction	ROC AUC	**1**	2	3
Australian	ROC AUC	3	2	**1**

These reversals show that the suitability of a fixed retrieval rule varies across tasks. Different rules allocate shared capacity to different candidate rows and query regions. To better understand the predictive value of these choices, we examine the label evidence carried by nearby context rows and its association with model loss.

F Post-Hoc Query Diagnostics

This complementary diagnostic examines 401 candidate shared contexts per task on 1,024 fixed validation queries at $B = 100$, covering Helena, Letter, MNIST, Volkert, and Connect-4. Table 9 reports the results. The analysis is intentionally post-hoc: the true query class defines "same class" after all candidate contexts and predictions are frozen. Entries in the final five columns are the fraction of context-induced query-level loss variation associated with each single diagnostic. We center loss within query, divide each diagnostic into 12 equal-frequency bins (or one bin per distinct value when there are fewer), and report the between-bin sum of squares divided by the total centered-loss sum of squares. To prevent class omission and the associated probability floor from dominating the

comparison, these associations are computed only over query–context observations in which the true class is present, with loss re-centered within query after this restriction. They should be read as descriptive associations, not causal effects.

Table 9: Query-level diagnostics at $B = 100$. $q = B/K$. "Missing" is the percentage of query–context observations whose true class is absent. The final five columns report associated loss variation (%). "True share" is the true class's share of reciprocal-rank evidence among selected rows.

Dataset	q	Missing	nearest any	nearest diff.	nearest same	true share	same count
Helena	1.00	23.0	1.9	0.2	11.7	10.8	12.0
Letter	3.85	2.0	12.5	0.5	31.3	29.5	4.6
MNIST	10.00	1.2	7.0	0.5	14.1	12.7	3.0
Volkert	10.00	0.7	2.4	0.1	14.4	12.5	5.2
Connect-4	33.33	0.0	0.8	2.6	5.5	9.8	24.7

Table 10: Held-out multivariate analysis of context-induced query loss. Generic R^2 uses only generic proximity variables. Augmented R^2 adds same-class and class-coverage variables.

Dataset	Generic R^2	Augmented R^2	ΔR^2
Helena	0.014	0.993	0.979
Letter	0.269	0.940	0.671
MNIST	0.253	0.909	0.656
Volkert	0.085	0.918	0.833
Connect-4	0.008	0.416	0.409

When the true class is absent, the predictor can assign it probability below 10^{-12}. The evaluation clips probabilities at 10^{-12}, producing a maximum per-query loss of $-\log 10^{-12} = 27.631$. The direction of the missing-class association is real, while its numerical magnitude depends on this necessary clipping convention.

G Additional Results

G.1 Main-ranking controls and confidence intervals

Tables 11 and 12 evaluate two additional method sets at $B = 100$. The first compares PAIR with five external selectors. The second adds the $\beta = 0$ control, which retains the retrieval configuration selected for full PAIR and disables the class-frequency correction. Each table recomputes ranks within its listed method set.

For all three tables, we first average each task's official metric over its three frozen outer splits, then rank the methods within that task, averaging ties. We compute percentile 95% confidence intervals from 100,000 bootstrap draws with seed 20260910. Each draw samples source datasets with replacement and retains all methods and tasks belonging to a sampled source. The two Diabetes tasks share one source and are sampled together. Each sampled task receives equal weight in the mean rank. The all-task, binary, and multiclass columns are resampled within their respective scopes. These intervals describe cross-dataset uncertainty conditional on the completed splits and validation choices. Table 2 uses the same procedure.

Table 11: Mean rank [95% confidence interval] at $B = 100$ among PAIR and five external selectors. The 51 tasks comprise 31 binary AUC and 20 multiclass log-loss tasks. Lower is better.

Method	All 51	Binary 31	Multiclass 20
PAIR	**1.58** [1.37, 1.80]	**1.71** [1.42, 2.03]	**1.38** [1.13, 1.65]
Facility Location	2.98 [2.67, 3.30]	3.55 [3.19, 3.90]	2.10 [1.75, 2.45]
Pooled kNN	3.52 [3.11, 3.94]	3.77 [3.19, 4.35]	3.13 [2.65, 3.63]
Uniform Random	3.76 [3.44, 4.08]	3.61 [3.13, 4.06]	4.00 [3.70, 4.30]
VIP-COp-lite-SR	4.02 [3.50, 4.51]	3.35 [2.71, 4.00]	5.05 [4.60, 5.45]
CRUMB (K=1)	5.14 [4.76, 5.48]	5.00 [4.45, 5.48]	5.35 [4.85, 5.75]

Table 12: Mean rank [95% confidence interval] at $B = 100$ among seven methods including the same-selector $\beta = 0$ control. This control uses the retrieval configuration selected for full PAIR. Lower is better.

Method	All 51	Binary 31	Multiclass 20
PAIR	**1.93** [1.72, 2.16]	**2.00** [1.69, 2.32]	**1.83** [1.53, 2.15]
PAIR ($\beta = 0$)	2.27 [2.02, 2.55]	2.48 [2.13, 2.87]	1.95 [1.68, 2.25]
Facility Location	3.80 [3.42, 4.18]	4.44 [4.02, 4.84]	2.83 [2.30, 3.35]
Pooled kNN	4.37 [3.90, 4.84]	4.61 [3.90, 5.29]	4.00 [3.40, 4.55]
Uniform Random	4.69 [4.32, 5.04]	4.48 [3.94, 5.00]	5.00 [4.70, 5.30]
VIP-COp-lite-SR	4.82 [4.22, 5.40]	4.03 [3.23, 4.81]	6.05 [5.60, 6.45]
CRUMB (K=1)	6.11 [5.71, 6.47]	5.95 [5.37, 6.45]	6.35 [5.85, 6.75]

G.2 Panel-wise rankings

Table 13 gives the full decomposition of the common-budget mean ranks for the method set in Table 12. The proposed method leads all three panels even though the leading baseline changes in each panel.

Table 13: Mean official-objective rank within each panel at $B = 100$, computed among the seven methods in Table 12. Lower is better.

Method	TabArena-23	Multiclass-13	Small-15
PAIR	**1.76**	**1.73**	**2.37**
PAIR ($\beta = 0$)	2.26	2.00	2.53
Facility Location	4.24	2.81	4.00
Pooled kNN	5.04	4.00	3.67
Uniform Random	4.48	5.08	4.67
VIP-COp-lite-SR	4.35	5.69	4.80
CRUMB ($K = 1$)	5.87	6.69	5.97

G.3 Backend-consistency results

Table 14 reports the complete $B = 100$ results for the four frozen predictors. VIP-COp-lite-SR searches directly under TabPFN-v3 and transfers the resulting row ranking to the other predictors, where it remains next-to-last.

Table 14: Mean rank at $B = 100$ across tasks supported by each frozen predictor: 51 for TabPFN-v3 and TabICL-v2, and 48 for TabFM and LimiX-16M. Official metrics are averaged over three splits before ranking the seven methods in Table 12 within each task. Lower is better.

Method	TabPFN-v3	TabICL-v2	TabFM	LimiX-16M
PAIR	**1.93**	**2.40**	**2.46**	**2.10**
PAIR ($\beta = 0$)	2.27	2.71	2.66	2.39
Facility Location	3.80	3.31	3.56	3.59
Pooled kNN	4.37	4.22	4.11	4.42
Uniform Random	4.69	4.39	4.44	4.58
VIP-COp-lite-SR (reference)	4.82	4.91	5.06	4.83
CRUMB ($K = 1$)	6.11	6.06	5.71	6.08

G.4 Regression panel

Table 15 reports the complete budget-wise mean RMSE ranks for the regression comparison in Section 5.5. The mean R^2 over cells is 0.682 for PAIR, 0.674 for Facility Location, 0.656 for Pooled kNN, 0.621 for Uniform Random, and 0.488 for CRUMB.

G.5 Matched-tuning controls

We compare the complete PAIR candidate set with a combination of simple retrieval rules chosen by validation. Each of Pooled kNN and Facility Location can select from five registered reservation

Table 15: Mean RMSE rank by budget on the 8-task regression panel (four predictors and five budgets, giving 160 dataset–predictor–budget cells). Lower is better.

Method	Overall	$B = 100$	$B = 200$	$B = 500$	$B = 1000$	$B = 2000$
PAIR	**1.70**	**1.45**	**1.75**	**1.72**	**1.81**	**1.78**
Facility Location	2.23	2.14	1.78	2.09	2.75	2.38
Pooled kNN	2.59	3.31	2.97	2.47	1.91	2.31
Uniform Random	3.56	3.19	3.53	3.72	3.53	3.81
CRUMB ($K = 1$)	4.92	4.91	4.97	5.00	5.00	4.72

fractions. Their union contains nine distinct context settings, each evaluated with $\beta \in \{0, 0.5, 1\}$, giving 27 scored configurations. PAIR uses its complete registered candidate set. Both methods select configurations on the same validation data before test evaluation. The $\beta = 0$ controls restrict the class-frequency correction during selection. "Selected β" allows validation to choose its strength.

For each task, we first average the test metric over three fixed splits. Table 16 ranks the ten methods by the official objective within each task and averages these ranks over the 51 tasks at $B = 100$. Binary tasks use AUC and multiclass tasks use log loss. Lower ranks are better.

Table 16: Matched-tuning controls at $B = 100$, 51 tasks. Methods are ranked by the official objective within each task and averaged. Lower is better.

Method	Mean rank
Full PAIR	**3.97**
Simple selector combination (selected β)	4.20
Reservation + Facility Location (selected β)	4.41
Simple selector combination ($\beta = 0$)	4.66
Reservation + Facility Location ($\beta = 0$)	5.03
Reservation + Pooled kNN (selected β)	5.42
Reservation + Pooled kNN ($\beta = 0$)	5.74
Class-Balanced Random	6.88
Facility Location (fixed)	7.06
Pooled kNN (fixed)	7.64

Results by task type. Table 17 compares PAIR with the simple selector combination with β chosen by validation. Mean ranks retain the ten methods of Table 16. Raw metric changes are aggregated within each task type. Positive improvement denotes higher AUC or lower log loss. Confidence intervals use 100,000 bootstrap samples drawn with replacement from tasks within each task type, after averaging the three splits. We report the 2.5th and 97.5th percentiles of the resulting distribution of mean improvements.

Table 17: Gains over the simple selector combination chosen by validation at $B = 100$. W/T/L counts PAIR wins, ties, and losses. Mean metrics are AUC for binary tasks and log loss for multiclass tasks. Positive improvement favors PAIR. Confidence intervals describe uncertainty in the mean.

Statistic	Binary (31 tasks)	Multiclass (20 tasks)
PAIR W/T/L	13/4/14	11/5/4
PAIR mean rank	4.613	2.975
Simple combination mean rank	4.419	3.850
PAIR mean metric	0.81177	0.97961
Simple combination mean metric	0.80823	1.00581
Mean improvement	0.00354	0.02620
95% interval	$[-0.00142, 0.00862]$	$[0.00157, 0.06148]$

Gain magnitude and sensitivity. Binary gains are asymmetric in magnitude. Across the 13 improved tasks, the mean and median AUC gains are 0.01570 and 0.01411. Across the 14 degraded tasks, the mean and median loss magnitudes are 0.00674 and 0.00377. Omitting each binary task in turn gives mean AUC gains between 0.00246 and 0.00488 over the remaining 30 tasks. The confidence interval for all 31 tasks includes zero, reflecting uncertainty in the estimated mean benefit.

Multiclass gains vary across tasks. Log loss decreases by 0.3049 on Letter, 0.0916 on Fashion-MNIST, and 0.0791 on Covertype. It increases by 0.0513 on Glass. Omitting Letter leaves 19 tasks with 10 wins, five ties, and four losses, and a mean reduction in log loss of 0.01153 with a 95% interval of $[-0.00147, 0.02630]$. Omitting the five tasks used to develop the diagnostics leaves 15 tasks with eight wins, three ties, and four losses, and a mean reduction of 0.01214 with a 95% interval of $[-0.00394, 0.03034]$. Table 18 gives every task's metric change, showing the distribution of benefits and declines.

Table 18: Gains of PAIR over the simple selector combination chosen by validation for each task at $B = 100$, after averaging three splits. Positive values indicate AUC increases for binary tasks and reductions in log loss for multiclass tasks. Each task type is sorted independently by improvement.

Binary task	AUC gain	Multiclass task	Log loss reduction
Amazon Employee Access	$+0.03604$	Letter	$+0.30491$
Credit G	$+0.03427$	Fashion-MNIST	$+0.09163$
HR Analytics	$+0.03146$	Covertype	$+0.07909$
Diabetes	$+0.02145$	Students Dropout	$+0.04232$
GiveMeSomeCredit	$+0.01756$	MNIST	$+0.03293$
Kc1	$+0.01615$	Waveform 5000	$+0.01274$
Heart Statlog	$+0.01411$	Jannis	$+0.01186$
Airline Satisfaction	$+0.01128$	Diabetes 130-US	$+0.00433$
Blood Transfusion	$+0.00703$	Vehicle	$+0.00419$
Taiwanese Bankruptcy	$+0.00688$	Volkert	$+0.00405$
Diabetes 130-US	$+0.00532$	Jungle Chess	$+0.00338$
Breast W	$+0.00153$	Connect-4	0
Online Shoppers Intention	$+0.00105$	Devanagari Script	0
Bank Marketing	0	Dilbert	0
In-Vehicle Coupon	0	Helena	0
JM1	0	SDSS17	0
Sonar	0	OkCupid STEM	-0.00018
Spambase	-0.00079	Splice	-0.00544
NATICUSdroid	-0.00093	Segment	-0.01045
Bank Customer Churn	-0.00175	Glass	-0.05134
Wdbc	-0.00183		
Ionosphere	-0.00247		
E-Commerce Shipping	-0.00262		
Credit Card Default	-0.00269		
Churn	-0.00485		
Australian	-0.00581		
HELOC	-0.00653		
Hazelnut Spread	-0.00743		
Fitness Club	-0.00934		
Marketing Campaign	-0.01062		
COIL 2000	-0.03667		

G.6 Paired official-objective comparisons

Table 19 reports the paired comparisons between PAIR and each other method over the 51 tasks at $B = 100$. A task is counted as a win for PAIR when its official-objective rank is strictly better than the row method's rank.

Table 19: Paired official-objective comparisons at $B = 100$, 51 tasks. W/T/L reports PAIR wins / ties / losses against the row method.

Method	Wins	Ties	Losses
PAIR ($\beta = 0$)	22	22	7
Facility Location	44	0	7
Pooled kNN	43	1	7
Uniform Random	47	0	4
VIP-COp-lite-SR	41	0	10
CRUMB ($K = 1$)	50	0	1

G.7 BUDGET-SCALING RANKS

Table 20 reports the complete rank comparison on the 29-dataset subset shared by four frozen predictors and all five budgets. On the 19 datasets where it is feasible, Exact Full Context attains mean rank 1.11 and wins 372 of 380 dataset–predictor–budget cells. At $B = 2000$, PAIR reaches 0.8372 AUC and 0.3458 log loss, versus 0.8508 and 0.3141 for Full Context.

Table 20: Mean rank by budget among six selectors on 29 datasets shared by four frozen predictors. Each dataset–predictor score is averaged over three splits before ranking. Lower is better.

Method	$B = 200$	$B = 500$	$B = 1000$	$B = 2000$
PAIR	**1.70**	**2.34**	**2.16**	**2.63**
Pooled kNN	2.96	2.64	2.83	2.87
Facility Location	2.83	2.86	2.80	2.95
Uniform Random	3.42	3.37	3.26	2.99
VIP-COp-lite-SR (reference)	4.77	4.94	5.53	5.60
CRUMB ($K = 1$)	5.33	4.85	4.41	3.97

G.8 ADDITIONAL BUDGET CELLS

Table 21 reports the budget cells that are not part of the 29-dataset complete-case subset. On the 36-task TabArena+Multiclass panel pooled over $B = 100$ and $B = 200$ (72 task–budget cells), the proposed method remains first. On Small-15, the proposed method leads at $B = 100$ and its $\beta = 0$ variant leads at $B = 64$ and $B = 128$, showing where the label correction becomes less stable on small validation sets.

Table 21: Mean official-objective rank at additional budgets. "Best baseline" is selected after evaluating the four existing selectors and is shown only to locate the proposed family relative to its strongest comparator.

Scope	Cells	PAIR	$\beta = 0$	Best baseline
TabArena+MC, $B = 100, 200$	72	**1.94**	2.32	Fac. Loc.: 3.54
Small-15, $B = 64$	15	2.70	**2.17**	Fac. Loc.: 2.60
Small-15, $B = 100$	15	**2.10**	2.27	Pooled kNN: 3.33
Small-15, $B = 128$	15	2.77	**2.50**	Fac. Loc.: 3.20

G.9 RESTRICTED-FAMILY ABLATIONS

Table 22 expands Table 3 with the mean rank of every restricted family and with the two β-free sensitivity rows. The two residual families pin every dimension to a baseline corner and leave only the reservation fraction free, so they are class-aware variants of Pooled kNN and Facility Location rather than pure class reservation. Mean ranks are computed among the nine rows below and therefore depend on that row set. The paired counts of Table 3 do not. The $\alpha = 0$ family retains pure class-reserved candidates ($f = 1$), for which no residual budget remains and α is inactive.

Two measurement notes accompany these comparisons. First, the model's outputs for omitted classes are constrained by class support, and the numerical loss also depends on probability clipping. We therefore assess class reservation using both class-absence rates and predictive outcomes after configuration re-selection. Second, the $\beta = 0$ comparison in Table 12 retains the retrieval configuration chosen with the correction enabled and measures the effect of disabling it at that configuration, whereas the re-selected control in Table 3 also allows the retrieval configuration to change.

G.10 FIXED-PARAMETER ALPHA SENSITIVITY

We vary $\alpha \in \{0, 0.5, 1, 2\}$ on five development tasks with frozen TabPFN-v3 at $B = 100$, fixing $f = 0.25$, $p = 2$, rank similarity, $\beta = 0$, and the class-reserved samples and seeds. Figure 6 reports each task's mean log-loss change from $\alpha = 0$, averaged over three frozen splits. A modest setting, $\alpha = 0.5$, reduces mean loss on all five tasks. Larger values produce task-dependent responses: $\alpha = 2$

Table 22: Restricted-family ablations at $B = 100$ across 51 tasks, after re-running validation selection inside each family. Mean rank is taken among these nine rows and lower is better. The mean gap is the official-objective utility of the full rule minus the variant. It pools ROC-AUC differences on binary tasks with log-loss differences on multiclass tasks.

Restricted family	Mean rank	Mean gap
Full unrestricted PAIR	**3.66**	—
No fairness weighting ($\alpha = 0$)	4.04	0.0015
No correction ($\beta = 0$, re-selected)	4.45	0.0018
Reservation + Facility Location residual (β free)	4.63	0.0970
Fixed accumulation order ($p = 1$)	5.16	0.0080
Reservation + Facility Location residual	5.26	0.0982
No class reservation ($f = 0$)	5.71	0.1249
Reservation + Pooled kNN residual (β free)	5.87	0.1051
Reservation + Pooled kNN residual	6.23	0.1065

lowers loss on Helena and raises it on the other four tasks. The sweep shows how the strength of coverage allocation changes predictive outcomes within a fixed retrieval configuration.

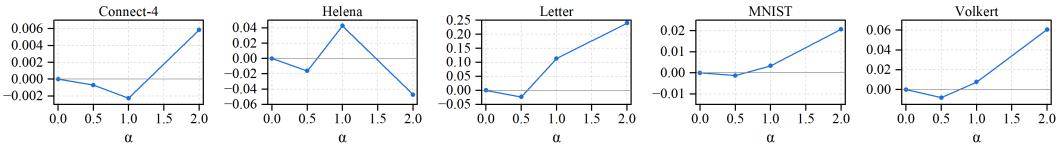

Figure 6: Fixed-parameter sensitivity to α on five development tasks. The vertical coordinate is mean log loss at α minus mean log loss at $\alpha = 0$, averaged over three splits. Negative values indicate improvement. Dashed lines mark zero change. Each panel uses its own vertical scale. All four evaluated values of α are shown.

G.11 Interior-point development screen

Before the 51-task outer-test aggregation, we used validation scores from five multiclass datasets to check whether the interior of the coverage family adds value beyond its baseline corners. The best interior setting improves over the best corner by 2.82% on Connect-4, 5.57% on Letter, 14.50% on MNIST, and 2.41% on Volkert. No interior point improves on pure class reservation for Helena, where $B = K = 100$. These are validation-space diagnostics, not outer-test claims. Their purpose is to establish that p, α, and f provide useful, non-equivalent interpolation among the baseline corners.

G.12 Query-batch regrouping sensitivity

On 12 tasks spanning binary to 100-class problems and original query batches of 16 or 256 rows, we compare the original frozen grouping with five deterministic random regroupings and one feature-structured regrouping. The feature-structured condition fits KMeans on test-query features standardized by training-fitted statistics only, orders queries by cluster and distance to its center, and slices the original batch-size sequence. All conditions reuse the same validation-selected PAIR selector, β, and frozen selector seed, so only batch membership changes. Test labels are never used for grouping or selection.

Table 23: Sensitivity of the aggregate task objective to query-batch grouping on 12 tasks. Entries are task-bootstrap means. Objective $|\Delta|$ mixes ROC-AUC differences on binary tasks and log-loss differences on multiclass tasks.

Regrouping	Objective $\lvert\Delta\rvert$
Random, 5 seeds	0.0201
Feature-structured KMeans	0.0768

Random regrouping changes the task objective by only 0.0201 on average, with six task means improving and six worsening after aligning the sign of ROC-AUC and log loss. The robustness we

report is at the level of the aggregate task objective. Selected contexts and individual predictions vary with grouping. Feature-structured grouping is a stronger intervention: objective $|\Delta|$ rises to 0.0768, and the effect can be beneficial or harmful depending on the task, with log-loss changes of -0.245 on Letter, -0.164 on Dilbert, and $+0.112$ on MNIST.

H Runtime and Memory

H.1 Complete-test-set inference protocol

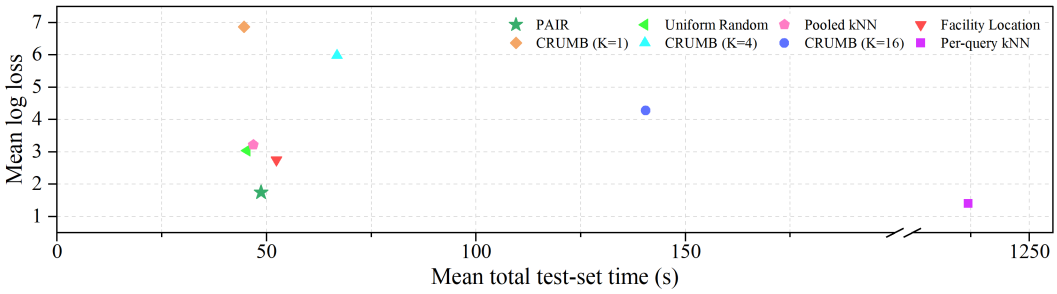

Figure 7: Complete-test-set inference time and predictive quality on Connect-4, Helena, Letter, MNIST, and Volkert. The eight methods each contribute one point, averaging three splits per dataset and then five datasets equally. Lower values are better on both axes. Time uses a linear scale with an axis break before per-query kNN.

Figure 7 compares eight retrieval methods on Connect-4, Helena, Letter, MNIST, and Volkert, with three frozen splits per task. The shared-context methods are PAIR, Uniform Random, Pooled kNN, Facility Location, and CRUMB ($K = 1$). Every selected context contains 100 distinct training rows. CRUMB forms 4 or 16 clusters inside each original query batch, normally containing 256 queries. Per-query kNN retrieves one context for each query. The predictor is frozen one-estimator TabPFN-v3. All methods use the same parallel query-context backend, configured with eight workers on one GPU and four CPU cores per GPU, with fewer workers on memory retries.

Total time measures the complete test set and includes context selection, worker and model startup, preprocessing, inference, and probability processing. Frozen dataset loading, result serialization, and the earlier validation search lie outside the timed interval. Each plotted point first averages three splits within each task, then averages the five tasks equally. The reported speedup is the ratio of these mean times.

H.2 Selection time and predictive rank

Figure 8 reports the selector contribution to runtime on 29 datasets at three large budgets. PAIR has the best mean within-budget rank at each budget, with selection time comparable to Facility Location. The model-aware VIP-COp-lite-SR selector has the largest measured selection time.

H.3 Controlled selector benchmark

Table 24 reports the controlled selector benchmark at the fixed point $N = 10{,}000$, $M = 64$, $B = 100$, and $D = 512$. All measurements use the deployed selector code with four CPU threads and no GPU.

Table 24: Selector CPU time and peak memory at the controlled fixed point. Selector-only time excludes the shared neighbor-cache construction, and cold-start includes it. Lower is better for time and memory.

Selector	Selector-only (s)	Cold-start (s)	Peak RSS (MiB)
Pooled kNN	0.003	0.131	205
Facility Location	0.379	0.508	206
PAIR interior	0.636	0.766	206

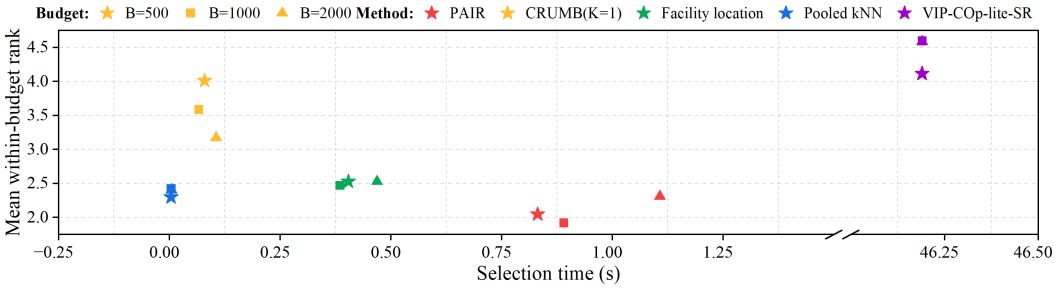

Figure 8: Selection time and predictive rank on 29 datasets at $B \in \{500, 1000, 2000\}$, with three splits per dataset. Points show mean within-budget rank against selection time in seconds on a linear scale with an axis break before VIP-COp-lite-SR. Lower values are better on both axes. Colors identify methods and shapes identify budgets. VIP-COp-lite-SR uses GPU wall-clock time and the other methods use CPU time.

Full validation for PAIR evaluates 78 distinct contexts and 234 scored prediction rules per dataset and budget, compared with the 1,920 model evaluations that VIP-COp-lite-SR uses per dataset and split. On the LimiX regression pipeline, batching 16 contexts per group with 4–5 concurrent slots per GPU improves throughput by 2.65–3.02× at 19.7–24.6 GiB peak GPU memory with unchanged predictions.

I Reproducibility Checklist

The supplementary experiment tree contains frozen protocol JSON files, resolved split and dataset manifests, context row ids, validation choices, raw per-split metrics, and aggregation scripts. Every reported main-table number can be reconstructed from the per-task official-objective CSV files by averaging the three outer splits and ranking methods within task. The primary 51-task summary, additional-budget summary, predictor-consistency summary, regression summary, matched-tuning summary, restricted-ablation summary, batch-composition summary, query-regroup summary, and runtime summary are stored separately so that unavailable budget cells are never silently mixed into the common-budget comparison.

AI USE STATEMENT

Generative AI tools were used as assistants for literature organization, code implementation and refactoring, analysis checks, and drafting and editing the manuscript. The authors inspected the cited sources, reviewed the resulting code and prose, and verified reported values against frozen experimental artifacts. The authors take responsibility for the study design, results, claims, and final text.

ETHICS STATEMENT

This work uses public benchmark datasets and introduces no new data collection or human-subject intervention. Some benchmark tables originate from sensitive application domains. We use them only to study a methodological retrieval question and draw no domain conclusions about the represented populations. "Fairness" in the fairness exponent denotes allocation of coverage across queries. It does not assert demographic fairness. Any deployment in a sensitive decision process would require a separate analysis of data provenance, group performance, calibration, and the consequences of changing context class proportions.

REPRODUCIBILITY STATEMENT

Section 5.1 and Appendix C specify the splits, query-batch constraints, budgets, predictor settings, selector grid, validation policy, and metrics. The supplementary artifacts include frozen manifests, selected context ids, validation selections, per-split outputs, and aggregation scripts. Appendix I describes how the reported rankings are reconstructed.