TABMOC: SCALING TABULAR FOUNDATION MODELS TO LARGE DATASETS VIA MIXTURE-OF-CONTEXTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) have emerged as strong predictors for diverse tabular tasks, yet their inference is typically limited to relatively small datasets. This limitation arises because TFMs condition on training data at inference time, making full-context inference increasingly memory-intensive as the training set grows. Existing approaches alleviate this cost by selecting only a small subset of the training data as context, which leave much of the available training information unused and therefore limit the predictive performance of TFMs. To address this issue, we investigate large-scale tabular prediction with TFMs through Mixture-of-Contexts (TabMoC). TabMoC partitions the training set into multiple small chunks and processes them separately with a tabular foundation model. A learned router then integrates these contexts across chunks to produce the final prediction. In this way, TabMoC respects the context-size constraint while drawing on information beyond a single context. Experiments demonstrate that TabMoC more closely approaches the performance of full-context inference, indicating that it can better exploit the potential of TFMs on large datasets. Moreover, as a compatible method, TabMoC consistently improves performance across multiple TFMs on both synthetic and real-world datasets.

1 INTRODUCTION

Tabular foundation models (TFMs) have emerged as a powerful approach to learning from structured data. Through pretraining across diverse tabular tasks, they learn to make predictions on new datasets by conditioning on labeled examples at inference time, without fitting a new set of model weights for each dataset (Müller et al., 2021; Hollmann et al., 2025). Recent models have achieved strong performance on classification and regression benchmarks, matching or surpassing gradient-boosted tree methods across a range of settings (Hollmann et al., 2025; Zhang et al., 2025; Qu et al., 2025). Their capabilities have also expanded beyond supervised prediction to include missing-value imputation (Zhang et al., 2025) and causal inference (Robertson et al., 2026). These advances demonstrate the potential of pretrained models to provide reusable predictive capabilities across heterogeneous tabular datasets and tasks.

In real-world applications, tabular datasets can contain millions or even billions of rows. Applying TFMs at this scale presents two challenges: memory constraints and long-context generalization. First, TFMs typically adopt Transformer architectures (Vaswani et al., 2017) to condition predictions on labeled training examples (Hollmann et al., 2025; Zhang et al., 2025; Qu et al., 2025), which encode tabular inputs into a sequence of tokens. As more training examples are included in the context, the computational and memory costs increase, which can make full-context prediction infeasible within a limited GPU memory. Second, TFMs are typically pretrained on large collections of tables, each containing a limited number of samples. Even when sufficient memory is available, conditioning on an entire large training table may require context lengths far beyond those encountered during pretraining. This mismatch in context length introduces an additional generalization challenge and may degrade predictive performance.

Existing work addresses this issue through several directions. Some studies enable longer-context inference by improving model architectures (Qu et al., 2025) or reducing the memory cost of attention computation (Sergazinov & Yin, 2025). Other studies keep contexts bounded by retrieving relevant training examples for the given test dataset (Xu et al., 2025). However, Transformer-based

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.

TFMs remain subject to context-length constraints, while selecting only a subset of training examples may leave useful information in the remaining data unexploited. This motivates us to study how TFMs can make full use of large training datasets under a limited context length.

In this paper, we introduce **Mixture-of-Contexts** (TabMoC), a framework for scaling tabular foundation models to large datasets by aggregating predictions across bounded contexts. TabMoC partitions the entire training set into bounded contexts and combines predictions from all context experts using query-dependent weights. A learned router refines a prior based on query–context proximity, adjusting the mixture weights to account for the reliability of each expert's prediction. The router is trained across synthetic tasks and applied directly to downstream datasets without further parameter updates. By aggregating predictions across contexts, TabMoC incorporates information from the full training set while keeping every backbone call within the context bound.

We evaluate TabMoC with LimiX (Zhang et al., 2025), TabPFN (Grinsztajn et al., 2025), and TabICL (Qu et al., 2026) on synthetic and real-world datasets for classification and regression. Across the three backbones, TabMoC consistently improves predictive performance over single-context selection and uniform aggregation, with routers trained exclusively on synthetic datasets and applied without target-dataset adaptation. Ablation studies further examine the effects of per-context capacity and training-set size, showing that TabMoC provides larger gains under tighter context constraints and can benefit from additional training data.

Our main contributions are as follows:

- We introduce TabMoC, a framework that incorporates information from the entire training set by aggregating predictions across bounded contexts of a shared TFM. This allows the amount of training data used for prediction to exceed the capacity of an individual context.
- We develop a lightweight router that learns query-dependent corrections to a geometric prior. The router is trained on synthetic tasks and transferred to downstream datasets without further parameter updates.
- We demonstrate the effectiveness of TabMoC across three tabular foundation models on synthetic and real-world classification and regression tasks. Ablation studies characterize its predictive performance under varying per-context capacities and training-set sizes.

2 Related Work

2.1 Tabular Foundation Models

Prior-data Fitted Networks (PFNs) learn to approximate Bayesian posterior prediction by pretraining on synthetic tasks sampled from a prior, enabling predictions on new datasets by conditioning on labeled examples without dataset-specific parameter updates (Müller et al., 2021). Building on this paradigm, TabPFN demonstrates strong performance on both classification and regression for small tabular datasets, using a transformer architecture that captures dependencies across rows and columns (Hollmann et al., 2025; Grinsztajn et al., 2025; 2026). Subsequent models have advanced both scalability and task coverage. TabICL introduces a two-stage architecture that first constructs fixed-dimensional row representations and then performs in-context prediction, enabling efficient classification on large tables (Qu et al., 2025). LimiX supports multiple tabular tasks, including classification, regression and missing-value imputation, within a single pretrained model (Zhang et al., 2025; 2026). Do-PFN (Robertson et al., 2026), StablePFN (Guan et al., 2026) and CausalPFN (Balazadeh et al., 2025) extend the PFN framework to causal effect estimation. Related extensions illustrate the broader applicability of this paradigm. PFNs4BO uses PFNs as surrogate models for Bayesian optimization (Müller et al., 2023). TabPFN-TS adapts tabular regression to time-series forecasting through lightweight feature engineering (Hoo et al., 2025).

2.2 Long-Context Inference

Long-context inference remains a major challenge for Transformers, which encode inputs into tokens and use attention to model interactions within a sequence (Vaswani et al., 2017; Li et al., 2024; Xiong et al., 2024; Bai et al., 2025). For dense attention over L tokens, computation scales quadratically with L, and explicitly storing the attention matrix requires $\mathcal{O}(L^2)$ memory. Memory-efficient implementations such as FlashAttention avoid storing the full matrix but retain quadratic

computation (Dao et al., 2022). Tabular foundation models such as TabPFN and TabICL inherit this limitation from their Transformer backbones (Hollmann et al., 2025; Qu et al., 2025).

Recent work has sought to accommodate larger training contexts through architectural and inference improvements. TabICL encodes each row into a fixed-dimensional representation before performing dataset-level in-context learning, reducing the cost of attention across rows (Qu et al., 2025). TabICLv2 introduces query- and context-length-dependent attention scaling to improve generalization to longer contexts (Qu et al., 2026). These advances enable larger contexts, but the full-context inference on very large tables can still exceed available memory capacity. Retrieval-based methods instead limit context size by selecting training examples relevant to test queries. LoCalPFN fine-tunes TabPFN on local training subsets and uses the nearest training examples as context for each test query (Thomas et al., 2024). Although context selection can draw on the full training set, the examples directly supplied to the model remain restricted to selected subsets. In this work, we investigate how to exploit information from the entire training set under a bounded context length.

2.3 Mixtures and Routing for Tabular Prediction

Mixture architectures combine multiple experts to capture different patterns in the data (Zhou et al., 2022; Yadav et al., 2024; Cai et al., 2025; Mu & Lin, 2025). Mixture-of-experts (MoE) models use an input-dependent gate to combine predictions from multiple experts (Jacobs et al., 1991). Sparse MoE scales language models by activating only a small subset of experts for each token (Shazeer et al., 2017). Switch Transformers simplify this routing to a single feed-forward expert per token (Fedus et al., 2022). In multimodal learning, Mixture-of-Transformers (MoT) uses separate parameters for different modalities while preserving global self-attention across their tokens (Liang et al., 2024). MoE and MoT achieve expert specialization through different sets of model parameters (Dai et al., 2024; Krishnamurthy et al., 2023). In tabular in-context learning, the labeled training context provides another way to define experts. Subsets of rows share the same feature schema and prediction task, but can capture different local patterns, making them natural sources of complementary predictions. This motivates us to explore the mechanism of Mixture-of-Contexts.

3 Method

3.1 Problem Formulation

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ denote a labeled training set and $\mathcal{X}_{\text{test}} = \{\mathbf{x}_q^{\text{test}}\}_{q=1}^{M}$ a set of test inputs whose labels $\{y_q^{\text{test}}\}_{q=1}^{M}$ are unavailable at inference time. A tabular foundation model f_θ conditions on a labeled context $\mathcal{C} \subseteq \mathcal{D}$ to produce

$$\hat{y}_q^{\text{test}} = f_\theta(\mathcal{C}, \mathbf{x}_q^{\text{test}}), \tag{1}$$

where $\hat{y}_q^{\text{test}}$ is the predicted result, *e.g.*, the predicted class for classification or the predicted value for regression. Let m denote the maximum number of labeled rows that can be accommodated within the bounded context size. When $N > m$, a model call cannot contain all of $\mathcal{D}$. Our objective is to predict the test labels accurately while making use of the available training set and respecting this context constraint. Equivalently, we seek a predictor F whose outputs $\hat{y}_q^{\text{test}} = F(\mathcal{D}, \mathbf{x}_q^{\text{test}}; m)$ minimize $\frac{1}{M}\sum_{q=1}^{M} \ell(\hat{y}_q^{\text{test}}, y_q^{\text{test}})$, with every call to f_θ conditioned on no more than m rows; ℓ is a task-appropriate prediction loss. Here the test labels define the evaluation objective, which are not available to F at inference time.

3.2 Mixture of Context Experts

A tabular foundation model can make different predictions for the same query when given different labeled contexts. Different contexts may reveal different patterns in the training data, and a context that is useful for one query may be less informative for another. This motivates Mixture-of-Contexts, where experts share the same model but differ in the labeled examples they use. Combining their predictions allows information from across the training set to contribute without requiring all rows to fit in a single context. Figure 1 illustrates the framework.

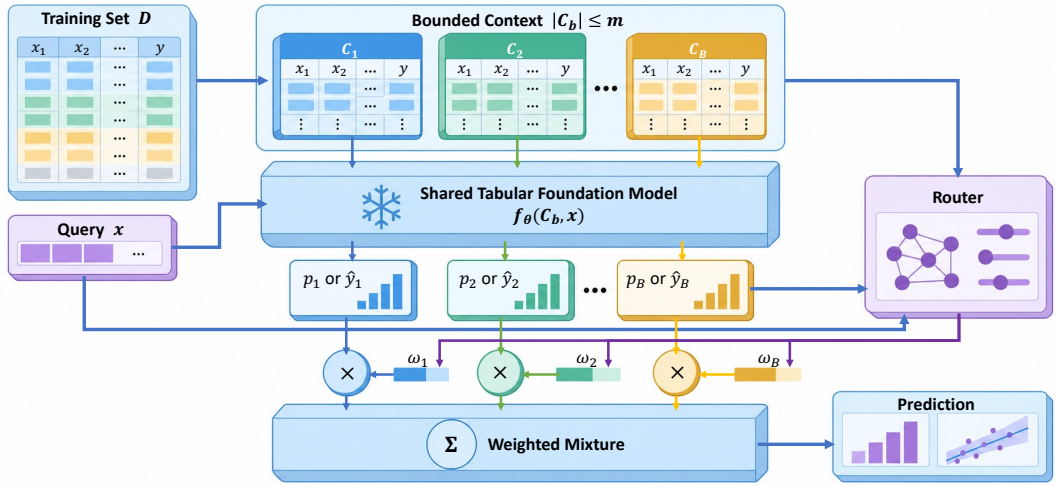

Figure 1: **Overview of Mixture-of-Contexts**. Different labeled contexts can reveal different predictive patterns to the same TFM, and their usefulness varies across queries. TabMoC combines the resulting predictions through learned, query-dependent weights, allowing information from across the training set to contribute without requiring all rows to fit in a single context. Each context contains at most m labeled rows. The router learns to aggregate these context experts across synthetic tasks and transfers to downstream datasets without further parameter updates.

To form contexts that capture local patterns in the training data, we group rows by their proximity in feature space. The resulting contexts form a partition $\{\mathcal{C}_b\}_{b=1}^{B}$ satisfying

$$\bigcup_{b=1}^{B} \mathcal{C}_b = \mathcal{D}, \qquad \mathcal{C}_b \cap \mathcal{C}_{b'} = \emptyset \text{ for all } b \neq b', \qquad |\mathcal{C}_b| \le m. \tag{2}$$

Every training row thus belongs to exactly one context, and each context stays within the model's capacity. In our implementation, we construct this partition using feature-space k-means and split oversized clusters into smaller contexts. The construction and preprocessing details are provided in Appendix A.1.

Each context $\mathcal{C}_b$ defines an expert using the shared model parameters θ. For a query $\mathbf{x}$, expert b produces a class-probability vector $\mathbf{p}_b(\mathbf{x}) = \mathbf{p}_\theta(\mathcal{C}_b, \mathbf{x})$ for classification or a scalar prediction $\hat{y}_b(\mathbf{x}) = f_\theta(\mathcal{C}_b, \mathbf{x})$ for regression. To account for differences in context relevance, we combine expert predictions using query-dependent weights $w_b(\mathbf{x}) \ge 0$ satisfying $\sum_{b=1}^{B} w_b(\mathbf{x}) = 1$. The mixture includes predictions from all context experts:

$$\begin{aligned} \mathbf{p}_{\text{TabMoC}}(\mathbf{x}) &= \sum_{b=1}^{B} w_b(\mathbf{x}) \mathbf{p}_b(\mathbf{x}), \quad \text{for classification,} \\ \hat{y}_{\text{TabMoC}}(\mathbf{x}) &= \sum_{b=1}^{B} w_b(\mathbf{x}) \hat{y}_b(\mathbf{x}), \quad \text{for regression.} \end{aligned} \tag{3}$$

Expert outputs are aggregated in a common prediction space, with class probabilities aligned across contexts and regression predictions expressed on a shared standardized target scale. The final prediction is obtained by taking the argmax of the mixed class probabilities or applying the inverse target transformation to the regression mixture. The query-dependent weights are determined by refining a geometric prior with a learned correction, which will be described in the following subsection.

3.3 Context Weighting with a Geometric Prior

Effective routing requires assessing both the relevance of a context to the query and the reliability of its expert prediction. Geometric proximity provides a useful starting point of the relevance,

but contexts at similar distances can still yield predictions of different quality. We therefore use geometric proximity to define prior weights for the contexts and learn to refine these weights based on expert predictions and context statistics.

The geometric prior reflects how closely a query matches the region represented by each context. We quantify the query's distance to context $\mathcal{C}_b$ in a common standardized feature space, denoted by $d_b(\mathbf{x})$. The precise definition is given in Appendix A.2. The prior assigns greater weight to closer contexts:

$$\pi_b(\mathbf{x}) = \frac{\exp(-d_b(\mathbf{x})/\tau)}{\sum_{j=1}^{B} \exp(-d_j(\mathbf{x})/\tau)}, \tag{4}$$

where $\tau > 0$ controls the concentration of the prior weights.

To refine the geometric prior, we introduce a learned router that evaluates the evidence provided by the context experts. Each query–context pair is represented by a set of fixed-dimensional descriptors $\mathbf{s}_b(\mathbf{x})$, to represent the expert predictions and context statistics. These descriptors are computed without the query labels. The router applies the same scoring network and aggregates information across contexts independently of their order. This design accommodates varying numbers of contexts while ensuring that reordering them leaves each context's weight unchanged. The signal definitions and network architecture are detailed in Appendices A.2 and A.3.

In implementation, the router with parameters ϕ predicts a correction $\delta_{\phi,b}(\mathbf{x})$ to the prior log-weight of each context. We bound the magnitude of each correction by $\rho > 0$ and obtain the normalized routing weights as

$$w_b(\mathbf{x}) = \frac{\pi_b(\mathbf{x}) \exp\big(\delta_{\phi,b}(\mathbf{x})\big)}{\sum_{j=1}^{B} \pi_j(\mathbf{x}) \exp\big(\delta_{\phi,j}(\mathbf{x})\big)}, \qquad |\delta_{\phi,b}(\mathbf{x})| \le \rho. \tag{5}$$

This parameterization recovers the geometric prior when all corrections are zero and allows the router to adjust the relative influence of contexts within a controlled range. Specifically, the ratio between any two context weights can increase or decrease by at most a factor of $\exp(2\rho)$, as shown in Appendix B.

3.4 Learning Across Synthetic Tasks

We train the router to assign context weights that improve the final prediction across tasks. We supervise the router on synthetic tasks using held-out query labels, optimizing the mixture predictions while keeping the foundation model frozen. The router parameters are shared across these tasks and remain fixed when applied to downstream datasets, enabling TabMoC to combine context experts on new tables without additional training.

We train the router through the loss of the final mixture prediction, supplemented by direct supervision of individual experts and a KL penalty on departures from the geometric prior. Averaging over the distribution P_{syn} of synthetic tasks, we optimize

$$\min_{\phi} \mathbb{E}_{\mathcal{T}\sim P_{\mathrm{syn}}} \left[\mathcal{L}_{\mathrm{mix}} + \lambda_{\mathrm{adv}}\mathcal{L}_{\mathrm{adv}} + \lambda_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}}\right], \tag{6}$$

where each loss is evaluated on the query set of $\mathcal{T}$, and $\lambda_{\mathrm{adv}}, \lambda_{\mathrm{KL}} \ge 0$ control the auxiliary terms. The mixture loss $\mathcal{L}_{\mathrm{mix}}$ optimizes the combined prediction using negative log-likelihood for classification and squared error for regression. The advantage loss $\mathcal{L}_{\mathrm{adv}}$ complements this aggregate supervision by assessing each expert's prediction relative to the mixture obtained under the geometric prior. For classification, the target measures an expert's relative improvement in true-class probability over the prior mixture. For regression, it measures the reduction in squared error relative to the prior mixture. The regularizer $\mathcal{L}_{\mathrm{KL}}$ penalizes deviations of the learned context weights from the geometric prior. For the precise loss function definitions and training details, we refer the readers to Appendix A.4.

At inference, TabMoC keeps both the backbone and router fixed and combines predictions from all contexts using query-dependent weights. This allows information from the full training set to contribute to the final prediction while keeping each context within the backbone's capacity. The complete inference procedure is summarized in Algorithm 1.

4 EXPERIMENTS

In this section, we present a comprehensive evaluation of TabMoC for tabular prediction. We first evaluate TabMoC on synthetic test datasets across three tabular foundation models, and then examine whether the learned routers transfer to real-world datasets without further training. We further analyze predictive performance under varying per-context capacities and training-set sizes. These experiments assess the effectiveness of TabMoC across classification and regression, with particular emphasis on the ability to benefit from training data beyond the capacity of a single context.

4.1 EXPERIMENTAL SETUP

Backbones and synthetic datasets. We use LimiX-16M (Zhang et al., 2025), TabPFN v2.5 (Grinsztajn et al., 2025), and TabICL v2 (Qu et al., 2026) as the backbones for classification and regression tasks. A separate router is trained for each backbone and prediction task. We construct synthetic datasets by first generating directed acyclic graphs to define variable dependencies and then sampling data from the corresponding generative processes. For each prediction task, we construct 100 datasets for training, 20 for validation, and 50 for testing, respectively. Under the default evaluation protocol, each test set contains 2,000 labeled training rows and 500 query rows. Details of synthetic data generation and router training are provided in Appendix C.

Real-world datasets. We evaluate on 80 real-world classification datasets and 50 real-world regression datasets. Their complete training splits contain 105–16,000 labeled rows for classification and 354–32,142 labeled rows for regression. Further details of the real-world datasets are provided in Appendix C.

Compared methods. We compare TabMoC with three baselines under the same context capability $m = 200$. *Random context* and *Nearest context* each use a single context per query, selected uniformly at random or by proximity to the context centroid, respectively. These baselines represent the general strategy of predicting from a selected context, as adopted by retrieval-based approaches (Thomas et al., 2024). *Uniform mixture* serves as a naive mixture baseline that assigns equal weight to all context predictions. *Full context* jointly conditions the tabular foundation model on all available training rows, providing an empirical reference for its performance ceiling without the context constraint. It uses the same training pool as TabMoC, allowing us to assess how closely aggregation across bounded contexts approaches this reference. For each dataset, prediction scores are averaged over three context partitions generated with different random seeds.

Metrics and aggregation. We evaluate classification performance using accuracy and ROC-AUC, with binary ROC-AUC for binary tasks and macro one-vs-one ROC-AUC for multiclass tasks. For regression, we report R^2 and normalized RMSE (nRMSE), defined as RMSE divided by the standard deviation of the training labels. Higher ROC-AUC, accuracy, and R^2 indicate better performance, whereas lower nRMSE is better. We first average scores over context partitions within each dataset and then compute an unweighted mean across datasets, ensuring that each dataset contributes equally to the final score.

4.2 RESULTS ON SYNTHETIC TASKS

We assess the generalization of TabMoC on synthetic test datasets that are disjoint from those used for router training and validation. Tables 1 and 2 report classification and regression performance across the three backbones, with the same context capability $m = 200$.

Classification. For classification, Uniform mixture outperforms Nearest context in AUC across the three backbones. TabMoC further improves mean AUC by 1.38–1.73 percentage points over Uniform mixture, achieving the best performance among the context-limited methods for every backbone. These gains narrow the gap to Full context without increasing the size of individual contexts, supporting query-dependent aggregation as a method to use the train set more effectively. The accuracy results support the same overall conclusion, with TabMoC achieving the highest mean accuracy among the context-limited methods for all three backbones and improving over Uniform mixture by 2.45–2.91 percentage points.

Regression. TabMoC achieves higher mean R^2 and lower mean nRMSE than both Nearest context and Uniform mixture across all three backbones. Its mean R^2 ranges from 0.333 to 0.342, corre-

Table 1: Classification on 50 synthetic test datasets. AUC and accuracy (Acc.) are percentages, averaged over partition seeds and then datasets. Bold marks the best mean among methods with context capability $m = 200$.

	LimiX		TabPFN		TabICL	
Method	AUC	Acc.	AUC	Acc.	AUC	Acc.
Random context	58.97	30.49	59.15	30.54	59.15	30.58
Nearest context	68.87	39.04	68.87	39.02	68.00	38.22
Uniform mixture	70.21	38.09	69.82	37.94	69.38	38.18
TabMoC (ours)	**71.59**	**41.00**	**71.40**	**40.80**	**71.11**	**40.63**
Full context	73.16	42.45	73.09	42.53	72.97	42.36

Table 2: Regression on 50 synthetic test datasets. Higher R^2 and lower nRMSE are better. Bold marks the best mean among methods with context capability $m = 200$.

	LimiX		TabPFN		TabICL	
Method	R^2	nRMSE	R^2	nRMSE	R^2	nRMSE
Random context	0.036	0.964	0.010	0.975	-0.100	1.008
Nearest context	0.326	0.795	0.324	0.796	0.288	0.815
Uniform mixture	0.252	0.846	0.251	0.846	0.251	0.844
TabMoC (ours)	**0.342**	**0.785**	**0.333**	**0.791**	**0.342**	**0.786**
Full context	0.401	0.745	0.398	0.747	0.398	0.747

sponding to improvements of 0.009–0.054 over Nearest context. These gains across both metrics demonstrate the effectiveness of query-dependent aggregation for the regression task under bounded context lengths.

4.3 Results on Real-World Tasks

We then evaluate whether the routers trained on synthetic tasks can generalize to real-world datasets without further parameter updates. Tables 3 and 4 summarize the results on classification and regression tasks respectively.

Classification. Across the 80 classification datasets, TabMoC achieves the highest mean AUC among the context-limited methods for all three backbones, improving over Uniform mixture by 1.57–1.90 percentage points. The accuracy results support a similar conclusion, with TabMoC achieving the highest mean accuracy among the context-limited methods for every backbone and exceeding Uniform mixture by 2.49–3.54 percentage points. These accuracy gains also narrow the gap to Full context. The consistent improvements across backbones and metrics support the transfer of the learned aggregation from synthetic tasks to real-world classification without further target-dataset adaptation.

Regression. Across the 50 regression datasets, TabMoC achieves the highest mean R^2 and lowest mean nRMSE among the context-limited methods for all three backbones. Compared with Nearest context, it improves mean R^2 by 0.033–0.040 and reduces mean nRMSE by 0.032–0.037. Its gains over Uniform mixture range from 0.301 to 0.328 in mean R^2. These improvements support query-dependent aggregation as an effective way to combine information across contexts, narrowing the gap to Full context while respecting the per-context constraint.

4.4 Ablation Studies

We conduct ablation studies with TabPFN to assess the effects of per-context capacity and training-set size on TabMoC's predictive performance. We first vary the per-context capacity while holding the total training-set size fixed. We then vary the training-set size under a fixed per-context capacity, examining whether access to more training data improves predictive performance.

Table 3: Classification on 80 real-world datasets. AUC and accuracy (Acc.) are percentages. Bold marks the best mean among methods with $m = 200$.

	LimiX		TabPFN		TabICL	
Method	AUC	Acc.	AUC	Acc.	AUC	Acc.
Random context	69.67	66.73	69.12	66.09	70.26	67.15
Nearest context	82.06	76.07	80.98	75.06	81.78	76.07
Uniform mixture	81.70	72.88	80.22	72.68	81.26	73.87
TabMoC (ours)	**83.28**	**76.43**	**82.12**	**75.57**	**83.05**	**76.36**
Full context	84.77	78.31	83.50	77.28	84.50	78.38

Table 4: Regression on 50 real-world datasets. Higher R^2 and lower nRMSE are better. Bold marks the best mean among methods with $m = 200$.

	LimiX		TabPFN		TabICL	
Method	R^2	nRMSE	R^2	nRMSE	R^2	nRMSE
Random context	-0.296	1.070	-0.584	1.119	-0.358	1.069
Nearest context	0.678	0.527	0.674	0.523	0.677	0.523
Uniform mixture	0.382	0.770	0.406	0.737	0.407	0.747
TabMoC (ours)	**0.711**	**0.494**	**0.707**	**0.491**	**0.717**	**0.486**
Full context	0.753	0.437	0.753	0.433	0.756	0.434

4.4.1 Ablation on Per-Context Capacity

We vary the maximum context size m while holding the training-set size fixed at 2,000 rows. For each prediction task, all configurations are evaluated on the same 50 synthetic test datasets. Each capacity uses a separately trained router selected on synthetic validation tasks.

As shown in Figure 2, increasing m improves both TabMoC and Uniform mixture, narrowing their performance gaps to the fixed Full context reference. TabMoC consistently outperforms Uniform mixture across all evaluated capacities. As m increases from 50 to 1000, its advantage decreases from 2.08 to 0.76 AUC percentage points for classification and from 0.127 to 0.045 in R^2 for regression. The larger gains at smaller capacities suggest that query-dependent aggregation is particularly valuable when each expert has access to fewer training examples.

4.4.2 Ablation on Training-Set Size

We vary the training-set size N while fixing the per-context capacity at $m = 200$. Nested training subsets are constructed from a common 5,000-row synthetic pool and evaluated on a fixed query set. A separate router is trained for each value of N.

As shown in Figure 3, TabMoC consistently outperforms Uniform mixture across all evaluated training-set sizes. From $N = 600$ to $N = 5000$, TabMoC's classification AUC increases from 69.88% to 71.56%, while its regression R^2 increases from 0.317 to 0.350. These results indicate that TabMoC can benefit from additional training data while maintaining a fixed context capacity for each backbone call.

5 Conclusion

We presented Mixture of Contexts (TabMoC), a framework that enables tabular foundation models to use the full training set through aggregation across bounded contexts. TabMoC defines experts through their conditioning data and combines their predictions using a learned router. The router is trained exclusively on synthetic tasks and transfers to downstream datasets without further parameter updates. Experiments with three backbones on synthetic and real-world classification and regression datasets demonstrate consistent improvements in predictive performance over single-context selection and uniform aggregation. The ablation results further demonstrate TabMoC's effectiveness

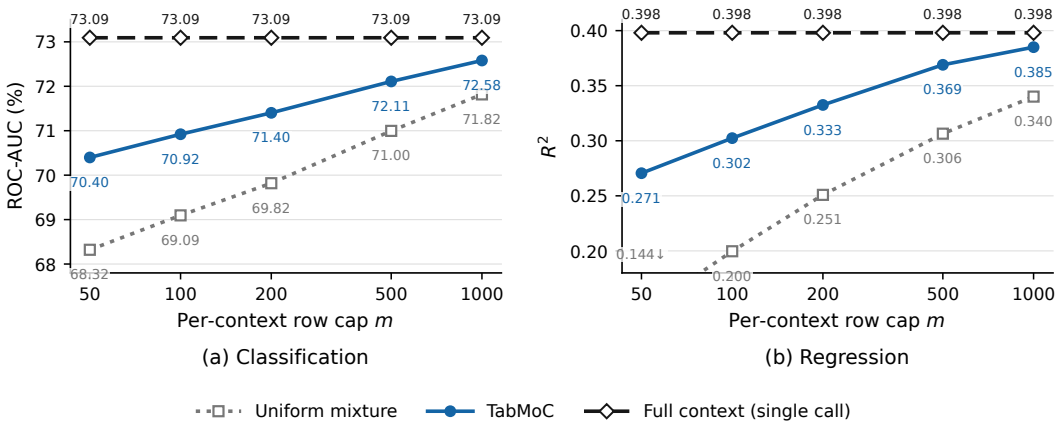

Figure 2: Effect of the per-context capability with TabPFN and a fixed pool of 2,000 training rows, evaluated on 50 test datasets per prediction task. Each value of m uses a separately trained router.

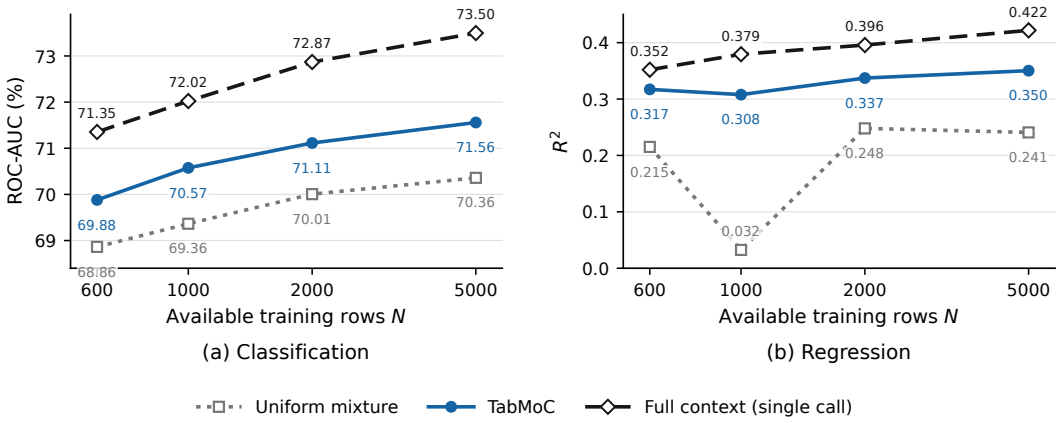

Figure 3: Effect of available training-set size with TabPFN at per-context capability $m = 200$. Each value of N uses a separately trained router.

under tight context constraints and its ability to benefit from larger training sets without increasing the per-context capacity. These findings highlight the value of combining predictions across contexts to extend the use of labeled data beyond what a tabular foundation model can accommodate in a single context.

AI USE STATEMENT

In this work, we used generative AI tools as a general-purpose writing assistant. Specifically, the LLM was employed to (1) check the grammar and correctness of the text and (2) suggest more natural and fluent wording. We also used generative AI tools to create graphical elements for the method overview figure. All AI-assisted text and graphical elements were reviewed and revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have taken several measures to ensure the reproducibility of our work. The experimental setup and evaluation protocols are described in Section 4.1, with detailed hyperparameters and implementation settings provided in Appendix C. To further facilitate reproducibility, we provide pseudocode for the TabMoC inference procedure in Algorithm 1.

REFERENCES

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Vahid Balazadeh, Hamidreza Kamkari, Valentin Thomas, Benson Li, Junwei Ma, Jesse C. Cresswell, and Rahul G. Krishnan. Causalpfn: Amortized causal effect estimation via in-context learning, 2025. URL https://arxiv.org/abs/2506.07918.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 37(7):3896–3915, 2025.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

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.

Zhengkang Guan, Yikang Chen, Haoyuan Qian, Kairong Han, Peng Cui, Fei Wu, and Kun Kuang. StablePFN: Stable Prediction with Causal-Aware Tabular Foundation Model. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1381–1392, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722592. doi: 10.1145/3770855.3818030. URL https://doi.org/10.1145/3770855.3818030.

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.

Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: Extending tabpfn-v2 to time series forecasting. *arXiv preprint arXiv:2501.02945*, 2025.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Yamuna Krishnamurthy, Chris Watkins, and Thomas Gaertner. Improving expert specialization in mixture of experts. *arXiv preprint arXiv:2302.14703*, 2023.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*, 2024.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications. *arXiv preprint arXiv:2503.07137*, 2025.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. Pfns4bo: In-context learning for bayesian optimization. In *International Conference on Machine Learning*, pp. 25444–25470. PMLR, 2023.

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

Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. Do-pfn: In-context learning for causal effect estimation. *Advances in Neural Information Processing Systems*, 38:174811–174848, 2026.

Renat Sergazinov and Shao-An Yin. Chunked tabpfn: Exact training-free in-context learning for long-context tabular data. *arXiv preprint arXiv:2509.00326*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4643–4663, 2024.

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.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*, 2024.

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.

Xingxuan Zhang, Gang Ren, Hao Yuan, Hao Zou, Hongze Tan, Hui Wang, Jianhao Song, Jiansheng Li, Jiayao Zhang, Jinghan Zhang, et al. Limix-2: A contextual mechanism network towards general structured-data intelligence. *arXiv preprint arXiv:2609.17488*, 2026.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

The appendix provides the implementation details, mathematical properties, experimental settings, and additional ablations that complement the main paper:

- **Appendix A** describes implementation details, including context construction, router descriptors and architecture, training objectives, and the inference procedure.
- **Appendix B** discusses the properties of context weighting.
- **Appendix C** specifies data generation, hyperparameters, and training and evaluation protocols.
- **Appendix D** provides additional experimental results, and examines the contribution of router training through a comparison with Distance Routing.

A Implementation and Training Details

A.1 Context Construction and Preprocessing

We standardize the training features using statistics computed from the labeled training set $\mathcal{D}$ and apply the same transformation to all query inputs. The standardized features are used to construct contexts and compute query–context distances. During synthetic training, these statistics are computed only from the labeled training portion of each task, excluding the query inputs.

We construct the initial partition by applying k-means to the standardized features with $k = \lceil N/m \rceil$. Within any cluster containing more than m rows, we order the rows by their distance to the cluster centroid and divide them into consecutive contexts of at most m rows. Smaller clusters are retained without further splitting. The resulting contexts satisfy the coverage, disjointness, and capacity conditions in Equation 2. Their sizes may differ, and splitting can increase the number of contexts B beyond the initial number of clusters.

Expert outputs are expressed in a common prediction space before aggregation. For classification, all probability vectors are aligned to the class space defined by the training labels. For regression, we standardize targets using the mean and standard deviation of all labels in $\mathcal{D}$. Expert predictions and router descriptors use the shared target scale, and the final mixture prediction is transformed back to the original scale. For each synthetic task, we standardize both context and query targets using the mean and standard deviation of the labeled training targets.

A.2 Query–Context Descriptors

The router descriptors summarize geometric proximity, expert predictions, and properties of the labeled contexts. All components are available when predicting an unlabeled query. For a query $\mathbf{x}$ and context $\mathcal{C}_b$, let $\widetilde{\mathbf{x}}$ and $\widetilde{\mathbf{x}}_i$ denote features standardized using the training-set statistics in Appendix A.1. Let d be the feature dimension and $\boldsymbol{\mu}_b$ the centroid of the standardized context features. We compute

$$d_b^{\mathrm{cent}}(\mathbf{x}) = \frac{\|\widetilde{\mathbf{x}} - \boldsymbol{\mu}_b\|_2}{\sqrt{d}}, \tag{7}$$

$$d_b^{\mathrm{near}}(\mathbf{x}) = \frac{\min_{i \in \mathcal{C}_b} \|\widetilde{\mathbf{x}} - \widetilde{\mathbf{x}}_i\|_2}{\sqrt{d}}, \tag{8}$$

and define $d_b(\mathbf{x}) = \frac{1}{2}\big(d_b^{\mathrm{cent}}(\mathbf{x}) + d_b^{\mathrm{near}}(\mathbf{x})\big)$ for the geometric prior in Equation 4.

Shared descriptors. For both prediction tasks, $\mathbf{s}_b(\mathbf{x})$ includes the prior weight π_b, the two geometric distances, the logarithm of the context size, the context missing-value rate, and the logarithm of the feature dimension.

Classification descriptors. For classification, the router uses statistics describing predictive confidence, agreement with the prior mixture, and the class distribution within each context. Predictive confidence is summarized by the normalized entropy of $\mathbf{p}_b$, its largest class probability, and the difference between its two largest class probabilities. The prior mixture $\overline{\mathbf{p}} = \sum_j \pi_j \mathbf{p}_j$ provides a reference for comparing expert predictions. We compare $\mathbf{p}_b$ with this reference through their KL divergence and inner product. We also record the probability that expert b assigns to the highest-probability class under $\overline{\mathbf{p}}$ and whether the expert itself predicts that class. Finally, the entropy of the

context labels summarizes their class distribution, while class coverage indicates how many of the dataset's classes are represented in $\mathcal{C}_b$.

Regression descriptors. For regression, the router uses statistics describing each expert's deviation from the prior mixture and the target distribution within its context. Predictions and context targets are expressed on the same standardized scale. We define

$$\overline{y} = \sum_j \pi_j \hat{y}_j, \qquad r_b = \hat{y}_b - \overline{y}, \qquad \sigma_{\text{exp}} = \left(\sum_j \pi_j r_j^2 \right)^{1/2}, \tag{9}$$

where $\overline{y}$ is the prior-weighted mean prediction, r_b is the deviation of expert b from this mean, and σ_{exp} is the prior-weighted standard deviation of the expert predictions. The router receives r_b, $|r_b|$, and a clipped version of r_b^2 to describe the direction and magnitude of the deviation. It also receives $r_b / \max(\sigma_{\text{exp}}, \epsilon)$, which expresses this deviation relative to the spread of expert predictions, with $\epsilon > 0$ preventing division by zero. The spread σ_{exp} and the rank of r_b among experts provide additional information about their overall disagreement and relative ordering. Finally, the mean, standard deviation, and interquartile range of the targets in $\mathcal{C}_b$ describe their central tendency and variability. None of these descriptors uses the query label.

A.3 Router Architecture

The router scores each context using both its own descriptor and summaries of the full collection of context experts. We omit the dependence on $\mathbf{x}$ in the following equations. A shared encoder E_ϕ maps each descriptor to $\mathbf{h}_b = E_\phi(\mathbf{s}_b)$. We then pool the context representations using a prior-weighted mean and an elementwise maximum:

$$\overline{\mathbf{h}}_\pi = \sum_{b=1}^{B} \pi_b \mathbf{h}_b, \qquad \mathbf{h}_{\max} = \max_b \mathbf{h}_b, \tag{10}$$

A shared utility head produces the context score

$$u_b = U_\phi\big([\mathbf{h}_b; \overline{\mathbf{h}}_\pi; \mathbf{h}_{\max}]\big). \tag{11}$$

The encoder and utility head each have two layers with hidden dimension 64.

We center the utility under the geometric prior and apply a bounded transformation to obtain the correction in Equation 5:

$$a_b = u_b - \sum_{j=1}^{B} \pi_j u_j, \qquad \delta_{\phi,b} = \rho \tanh(a_b), \qquad \rho > 0. \tag{12}$$

Centering expresses each utility relative to the prior-weighted mean, while the scaled hyperbolic tangent bounds the magnitude of the log-weight correction. The final layer of the utility head is initialized to zero, so the router initially reproduces the geometric prior exactly.

A.4 Router Training Objective

Router training combines supervision of the final mixture prediction with an auxiliary assessment of individual experts and regularization toward the geometric prior. For a synthetic task with Q query pairs $(\mathbf{x}_q, y_q)$, write $\pi_{qb} = \pi_b(\mathbf{x}_q)$, $w_{qb} = w_b(\mathbf{x}_q)$, and $a_{qb} = a_b(\mathbf{x}_q)$. Let $\boldsymbol{\pi}_q$ and $\mathbf{w}_q$ denote the corresponding weight vectors.

Mixture loss. The primary term in Equation 6 evaluates the combined prediction using negative log-likelihood for classification or squared error for regression:

$$\mathcal{L}_{\text{mix}} = \begin{cases} -\dfrac{1}{Q} \displaystyle\sum_{q=1}^{Q} \log p_{\text{TabMoC}}(y_q \mid \mathbf{x}_q), & \text{for classification,} \\ \dfrac{1}{Q} \displaystyle\sum_{q=1}^{Q} \big(\hat{y}_{\text{TabMoC}}(\mathbf{x}_q) - y_q\big)^2, & \text{for regression,} \end{cases} \tag{13}$$

where $p_{\mathrm{TabMoC}}(y_q \mid \mathbf{x}_q)$ is the mixed probability assigned to the true class. Regression predictions and query labels use the common target scale fitted on the task's labeled training portion.

Expert advantage targets. The auxiliary targets measure how each expert compares with the prediction obtained under the geometric prior. For classification, let $p_b(y)$ and $\overline{p}(y)$ denote the probabilities assigned to class y by expert b and the prior mixture $\overline{\mathbf{p}} = \sum_j \pi_j \mathbf{p}_j$, respectively. We use

$$A_b^{\mathrm{cls}} = \mathrm{clip}\left(\frac{p_b(y)}{\max(\overline{p}(y), \epsilon)} - 1, -c, c\right), \tag{14}$$

which measures the expert's relative improvement in true-class probability. For regression, the target measures the reduction in squared error relative to $\overline{y} = \sum_j \pi_j \hat{y}_j$:

$$A_b^{\mathrm{reg}} = \mathrm{clip}\left((\overline{y} - y)^2 - (\hat{y}_b - y)^2, -c, c\right). \tag{15}$$

In both cases, $c > 0$ bounds the target through $\mathrm{clip}(v, -c, c) = \min(c, \max(-c, v))$. Synthetic query labels are used only to construct training supervision and are excluded from the expert inputs and router descriptors.

Auxiliary loss and regularization. Let A_{qb} denote the task-specific target for query q and expert b. The remaining terms in Equation 6 are

$$\mathcal{L}_{\mathrm{adv}} = \frac{1}{Q}\sum_{q=1}^{Q}\sum_{b=1}^{B} \pi_{qb}\,\mathrm{SmoothL1}(a_{qb} - A_{qb}), \tag{16}$$

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{Q}\sum_{q=1}^{Q} \mathrm{KL}(\mathbf{w}_q \| \boldsymbol{\pi}_q), \tag{17}$$

where $\mathrm{SmoothL1}$ is the unit-threshold smooth-L1 penalty and $\mathrm{KL}(\mathbf{w}_q \| \boldsymbol{\pi}_q) = \sum_b w_{qb} \log(w_{qb}/\pi_{qb})$. The advantage loss supervises the centered context utilities, while the KL term regularizes the learned weights toward the geometric prior. Only the router parameters are updated. Optimization settings and model selection are described in Appendix C.

A.5 Inference Procedure

Algorithm 1 summarizes inference with a frozen backbone and a trained router. The function G_ϕ denotes the trained router. Classification probabilities are retained when required for probability-based evaluation metrics.

B Properties of the Context Mixture

We examine the invariance and reweighting properties of TabMoC and the interpretation of its auxiliary supervision.

Invariance to context ordering. Permuting the contexts permutes their descriptors and prior weights in the same way. The shared encoder and utility head apply the same functions to each context, while both pooling operations in Equation 10 remain unchanged. The utilities, corrections, and mixture weights therefore follow the same permutation as the expert predictions. Consequently, the weighted prediction in Equation 3 is invariant to context ordering. The shared scoring functions and pooling operations also allow the architecture to accommodate different numbers of contexts.

Bounded changes in relative context weights. For finite distances, the geometric prior assigns positive weight to every context. The normalizing factors in Equation 5 cancel when taking the ratio between two context weights, giving

$$\frac{w_b/w_j}{\pi_b/\pi_j} = \exp(\delta_{\phi,b} - \delta_{\phi,j}) \in [\exp(-2\rho), \exp(2\rho)]. \tag{18}$$

The interval follows from $|\delta_{\phi,b}|, |\delta_{\phi,j}| \le \rho$. Thus, the relative weight of any two contexts can increase or decrease by at most a factor of $\exp(2\rho)$ compared with the prior. Finite corrections preserve positive mixture weights, so all context experts remain represented in the prediction.

Algorithm 1 TabMoC inference

Require: Training set $\mathcal{D}$, test inputs $\mathcal{X}_{\text{test}}$, context capacity m, prior temperature τ, correction scale ρ, frozen backbone f_θ, trained router G_ϕ

Ensure: Final predictions $\{\hat{y}_q^{\text{test}}\}_{q=1}^{M}$

```
Fit feature preprocessing on D and construct {C_b}_{b=1}^B as in Appendix A.1
For regression, standardize context targets using all labels in D
for b = 1, ..., B do
    Obtain expert predictions for all test inputs from f_θ conditioned on C_b
end for
for each x_q^test ∈ X_test do
    Compute the geometric prior using Equation 4
    Construct {s_b(x_q^test)}_{b=1}^B as in Appendix A.2
    Apply G_φ and Equation 5 to obtain the mixture weights
    Aggregate all expert predictions using Equation 3
    if classification then
        ŷ_q^test ← arg max_c [p_TabMoC(x_q^test)]_c
    else
        Inverse-transform ŷ_TabMoC(x_q^test) to obtain ŷ_q^test
    end if
end for
return {ŷ_q^test}_{q=1}^M and, for classification, the mixed probability vectors
```

C Experimental Settings and Hyperparameters

Table 5 summarizes the default training and evaluation settings, and Table 6 lists the router hyperparameters.

Synthetic data generation. We construct classification and regression datasets by sampling directed acyclic graphs and generating observations from the resulting dependencies. The generator uses 14–24 visible source variables and 395–580 hidden nodes, with a noise scale of 1.2 and a nuisance-feature ratio of 0.3. For classification, the number of classes is sampled from $\{2, 3, 4, 5, 6, 8, 10\}$. For each prediction task, we use 100 datasets for router training, 20 for validation, and 50 for testing. All 170 datasets have distinct generative rule configurations. Under the default test protocol, each synthetic test dataset contains 2,000 labeled training rows and 500 query rows.

Router training and model selection. For each synthetic training dataset, we generate a fixed table of 2,000 rows. At each of 20 training epochs, 256 rows are assigned to the query set and the remaining 1,744 rows form the labeled training portion. Epoch-specific random seeds vary the training/query split and the context partition while retaining the underlying table. Feature and target preprocessing follow Appendix A.1, and query labels enter only the training loss. A separate router is trained for each backbone and prediction task, with the backbone parameters held fixed.

Optimization. We optimize the router with AdamW using the settings in Table 6.

Real-world datasets. Following LimiX (Zhang et al., 2025), we use the provided classification and regression datasets and their official training/test splits. Classification is evaluated on 80 datasets using the complete training splits, which contain 105–16,000 labeled rows. For regression, we evaluate 50 datasets, which contain 354–32,142 training rows per dataset. The complete test split is retained in both tasks. We excluded some datasets provided by LimiX because evaluating them in full context exceeded our GPU memory. All compared methods use the same training rows and test inputs, and methods operating on bounded contexts share the same context partitions.

Score aggregation Scores are first averaged over the three context partitions within each dataset and then averaged with equal weight across datasets. This ensures that each dataset contributes equally to the final score.

Table 5: Default training and evaluation settings. Synthetic datasets are used to train and select the routers, which are then evaluated on real-world datasets without further parameter updates.

Setting	Classification	Regression
Synthetic datasets (training / validation / test)	100 / 20 / 50	100 / 20 / 50
Rows per router-training dataset	2,000	2,000
Training / query rows per epoch	1,744 / 256	1,744 / 256
Synthetic test (training / query rows)	2,000 / 500	2,000 / 500
Default context capacity	$m = 200$	$m = 200$
Context partition seeds	42, 43, 44	42, 43, 44
Primary metric	ROC-AUC ↑	R^2 ↑
Secondary metric	Accuracy ↑	nRMSE ↓
Model selection	Validation mixture NLL	Validation standardized mixture MSE
Real-world test datasets	80	50

Table 6: Default hyperparameters for router training. The backbone parameters remain frozen.

Hyperparameter	Value
Hidden dimension	64
Training epochs	20
Optimizer	AdamW
Learning rate	10^{-3}
Weight decay	10^{-4}
Prior temperature τ	0.25
Advantage loss weight λ_{adv}	1
KL loss weight λ_{KL}	0.01

D Ablation of Router Training

To assess the contribution of router training, we compare TabMoC with *Distance Routing*, which directly sets $w_b(\mathbf{x}) = \pi_b(\mathbf{x})$ without a learned correction. Both variants use the same context partitions and frozen expert predictions and aggregate predictions from every context. The comparison therefore isolates the effect of the learned correction.

D.1 Results across Backbones and Prediction Tasks

Table 7 reports mean AUC and R^2. On synthetic datasets, TabMoC improves both classification and regression performance for all three backbones. The gains on real-world classification are smaller, ranging from 0.06 to 0.07 AUC percentage points. On real-world regression, TabMoC improves mean R^2 by 0.013–0.027 for LimiX, TabPFN, and TabICL, respectively. These results indicate that learned weighting can improve upon geometric proximity.

Normalized regression error. The nRMSE results in Table 8 provide a complementary assessment of regression performance. TabMoC achieves lower mean nRMSE in all six settings, with reported reductions of 0.011–0.015 on synthetic datasets and 0.021–0.034 on real-world datasets. These comparisons report mean differences without confidence intervals.

D.2 Effect of Per-Context Capacity

Table 9 extends the context-capacity experiment with TabPFN by including Distance Routing. TabMoC improves mean performance at every evaluated capacity. As m increases from 50 to 1000, its advantage decreases from 0.34 to 0.05 AUC percentage points for classification and from approximately 0.036 to 0.005 in regression R^2. The larger gains at smaller capacities indicate that learning to adjust the geometric weights is particularly useful when each context contains fewer training examples.

Table 7: Effect of router training on classification AUC and regression R^2. Distance Routing uses the geometric prior without a learned correction. Δ denotes the mean paired difference between TabMoC and Distance Routing. AUC is reported as a percentage, and AUC differences are in percentage points.

Backbone	Distance Routing	TabMoC	Δ
Synthetic classification: 50 test datasets			
LimiX	71.39	71.59	$+0.21$
TabPFN	71.23	71.40	$+0.17$
TabICL	70.77	71.11	$+0.34$
Real-world classification: 80 test datasets			
LimiX	83.21	83.28	$+0.07$
TabPFN	82.06	82.12	$+0.06$
TabICL	83.00	83.05	$+0.06$
Synthetic regression: 50 test datasets			
LimiX	0.322	0.342	$+0.020$
TabPFN	0.317	0.333	$+0.016$
TabICL	0.322	0.342	$+0.020$
Real-world regression: 50 test datasets			
LimiX	0.684	0.711	$+0.027$
TabPFN	0.694	0.707	$+0.013$
TabICL	0.692	0.717	$+0.025$

Table 8: Effect of router training on nRMSE. Lower values are better. Δ denotes TabMoC minus Distance Routing, so negative differences favor TabMoC.

Dataset group	Backbone	Distance Routing	TabMoC	Δ nRMSE
Synthetic	LimiX	0.800	0.785	-0.015
Synthetic	TabPFN	0.803	0.791	-0.011
Synthetic	TabICL	0.799	0.786	-0.013
Real-world	LimiX	0.529	0.494	-0.034
Real-world	TabPFN	0.512	0.491	-0.021
Real-world	TabICL	0.519	0.486	-0.032

D.3 Effect of Training-Set Size

Table 10 reports the same comparison across training-set sizes with $m = 200$. TabMoC consistently improves on Distance Routing, with classification gains of 0.10–0.16 AUC percentage points. For regression, the R^2 improvement increases from approximately 0.013 at $N = 600$ to 0.037 at $N = 5000$. These results support learned aggregation across the evaluated training-set sizes, with larger regression gains as more training data become available.

Table 9: Effect of router training across context capacities with TabPFN and $N = 2000$. A separate router is trained for each capacity. Δ denotes TabMoC minus Distance Routing. AUC differences are in percentage points. Figure 2 presents the corresponding comparison with Uniform mixture and Full context.

Context capacity m	Distance Routing	TabMoC	Δ
Classification: AUC (%)			
50	70.06	70.40	+0.34
100	70.70	70.92	+0.22
200	71.23	71.40	+0.17
500	72.01	72.11	+0.10
1000	72.53	72.58	+0.05
Regression: R^2			
50	0.234	0.271	+0.036
100	0.278	0.302	+0.024
200	0.317	0.333	+0.016
500	0.360	0.369	+0.009
1000	0.380	0.385	+0.005

Table 10: Effect of router training across training-set sizes with TabPFN and $m = 200$. A separate router is trained for each value of N. Δ denotes TabMoC minus Distance Routing. AUC differences are in percentage points. Figure 3 presents the corresponding comparison with Uniform mixture and Full context.

Training-set size N	Distance Routing	TabMoC	Δ
Classification: AUC (%)			
600	69.77	69.88	+0.11
1000	70.47	70.57	+0.10
2000	70.97	71.11	+0.14
5000	71.40	71.56	+0.16
Regression: R^2			
600	0.304	0.317	+0.013
1000	0.293	0.308	+0.015
2000	0.317	0.337	+0.020
5000	0.313	0.350	+0.037