MT-TAB: LEVERAGING INTER-TARGET DEPENDENCIES IN TABULAR FOUNDATION MODELS FOR MULTI-TARGET PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular data are widely used across industrial domains such as manufacturing, finance, and healthcare. Tabular foundation models (TFMs), including TabPFN and TabICL, showcase strong generalization capabilities, outperforming Gradient Boosted Decision Trees (GBDTs) through in-context learning. However, existing TFMs are designed for *single-target* prediction. When applied to multi-target prediction problems which are common in real-world applications, they suffer from three limitations: (i) inference costs and resource requirements scale linearly with the number of targets, (ii) dependencies among targets are not fully leveraged, and (iii) directly modeling a joint distribution during pretraining is computationally expensive and increases architectural complexity. In particular, (ii) is the central motivation of this paper: single-target TFMs fundamentally exclude from the prediction path the information among multiple targets that is usable for prediction, which we call *target dependency*. Therefore, we propose MT-Tab, a multi-target Tabular foundation model framework that reformulates pretrained TFMs to tackle multi-target prediction problems. MT-Tab *unfolds* multi-target labels along the meta/batch dimension via *target unfolding*, passes them through the pretrained target encoder, and integrates them into the TFM's transformer input through *prompt expansion*. To preserve pretrained priors, we keep the TFM parameters frozen and employ Low-Rank Adaptation (LoRA) for efficient task adaptation. On the newly constructed *MT-TabBench* with 10 tasks, MT-Tab achieves the highest Elo rating among 17 methods including 16 baselines. Specifically, MT-Tab yields up to a 7.1% performance gain over the strongest single-target TFM under target imbalance by capturing target dependencies, while achieving up to $11.2\times$ faster inference in many targets.

1 INTRODUCTION

Tabular data often consist of numerical and categorical variables, and exhibit strong heterogeneity across domains in terms of feature semantics, distributions, and missing-value patterns (Borisov et al., 2022). Gradient Boosted Decision Trees (GBDTs) have long been dominant in tabular data analysis due to their inherent robustness to missing values, efficiency in training, and reliability on small-scale datasets (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022). Tabular foundation models (TFMs), such as TabPFN and TabICL, have emerged as effective alternatives to GBDTs through in-context learning on tabular data (Müller et al., 2021; Hollmann et al., 2025; Qu et al., 2025). Pretrained on synthetic tasks, these models transfer learned priors to new datasets using labeled examples as context, without requiring task-specific parameter updates.

Despite their strong performance, most TFMs are primarily designed for single-target prediction, where each task involves predicting a single-output variable, whereas many applications require predicting multiple targets from shared input features (Spyromitros-Xioufis et al., 2016). A straightforward extension of existing TFMs is to perform inference independently for each target, but this strategy requires repeated model execution. More importantly, treating each target as an isolated prediction task overlooks the statistical dependencies that may exist among multiple outputs.

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.

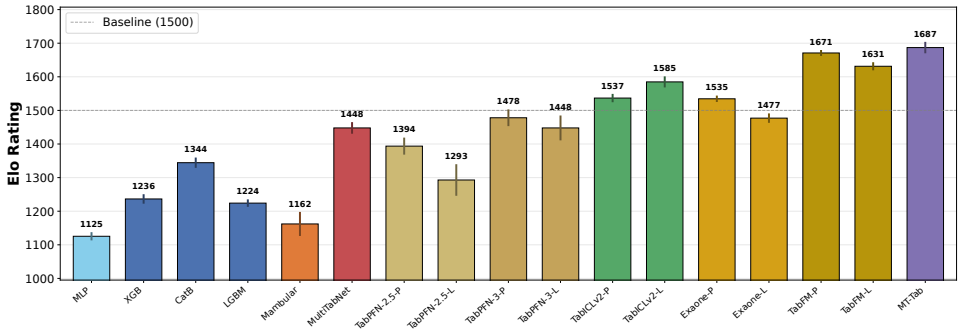

Figure 1: Performance comparison on MT-TabBench. The proposed MT-Tab achieves the highest rating. "P" denotes prompt evaluation and "L" denotes LoRA fine-tuning. The Elo rating computation is described in Appendix Sec. I.1.

Such target dependencies provide an important source of predictive information in multi-target learning. Multi-task learning exploits relatedness among targets through shared representations (Zhang & Yang, 2021; Ma et al., 2018), while approaches such as input-space expansion for multi-target regression and classifier chains explicitly condition the prediction of one target on other target variables (Spyromitros-Xioufis et al., 2016; Read et al., 2011). These approaches demonstrate that modeling target dependencies can provide information that is unavailable when targets are predicted independently. Performance of TFMs can deteriorate when the target-domain distribution differs substantially from the synthetic priors encountered during pretraining (Garg et al., 2025). Therefore, effective multi-target TFMs should not only exploit dependencies among targets, but also adapt to the target domain while preserving the generalization capabilities acquired during pretraining.

We propose MT-Tab, a framework that converts a pretrained single-target TFM into a multi-target predictor without additional pretraining. Through *target unfolding*, MT-Tab encodes each target using the pretrained single-target encoder. It then combines these target embeddings with shared input embeddings through *prompt expansion*, allowing self-attention to integrate observed support information across targets and predict all targets in a single forward pass. Low-Rank Adaptation (LoRA) (Hu et al., 2021) adapts the expanded representation to the target domain while keeping the pretrained parameters frozen.

Our main contributions are as follows.

- We introduce MT-Tab, a framework that extends a pretrained single-target TFM into a multi-target predictor without additional multi-target pretraining.
- We enable cross-target information sharing within the transformer through target unfolding and prompt expansion, allowing MT-Tab to exploit target dependencies. Our experiments demonstrate the benefits of this mechanism (Sec. 5.1).
- We introduce *MT-TabBench*, a benchmark curated from existing datasets, and conduct extensive experiments to evaluate both predictive performance (Fig. 1) and inference efficiency (Sec. 5.5).

2 Background

Tabular Foundation Models. Building on TabPFN and TabICL, recent TFMs have advanced predictive performance and scalability. TabPFN-2.5 (Grinsztajn et al., 2025) and TabPFN-3 (Grinsztajn et al., 2026) extend the TabPFN family, while TabICLv2 (Qu et al., 2026) further develops scalable in-context prediction. Other recent models include TabFM (Research, 2026) and Exaone-Tabular (Eo et al., 2026). Despite these advances, the single-target interfaces used by these models require separate prediction for each target when applied to multi-target tasks. Our work addresses this limitation by extending a pretrained single-target TFM to jointly process multiple targets.

Supervised Multi-target methods. MultiTab-Net (Sinodinos et al., 2026) and Mambular (Thielmann et al., 2024) support multi-target prediction using task-specific attention and a Mamba-based

architecture, respectively. In contrast, our method introduces target dependencies into an existing pretrained single-target TFM. Fig. 2 compares these approaches in the 5-shot setting. TabPFN-MT (Cureton & Armanfard, 2026) enables simultaneous multi-target prediction through multi-target synthetic pretraining, whereas MT-Tab reuses a single-target checkpoint without additional pretraining, with downstream adaptation taking at most approximately one hour on a single GPU across our benchmarks. Appendix Sec. C.2 compares the methods, training budgets, and scope of comparison.

Conditional Prediction Across Targets. Regressor chains, classifier chains, and NPE-PFN exploit target dependencies by conditioning each prediction on preceding targets (Spyromitros-Xioufis et al., 2016; Read et al., 2011; Vetter et al., 2026). These methods can use earlier target predictions as conditions for subsequent predictions at inference time. In contrast, MT-Tab integrates observed support targets into a shared representation and predicts all query targets simultaneously, without conditioning on preceding query-target predictions.

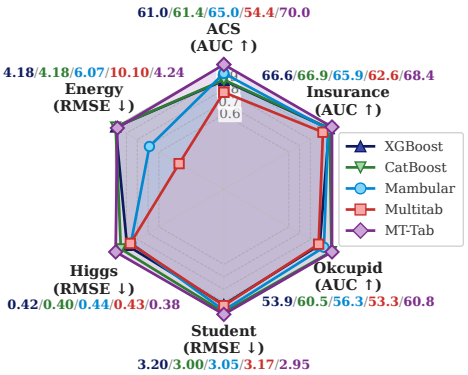

Figure 2: 5-shot performance comparison among multi-target models. Each axis is normalized across five models to $[0, 1]$ (higher is better; best $= 1$). MT-Tab outperforms all baselines on all datasets except Energy.

LoRA. Many Parameter-Efficient Fine-Tuning (PEFT) methods have been proposed to adapt pretrained models to specific domains. Among them, Hu et al. (2021) freezes pretrained weights and learns low-rank updates parameterized by two trainable matrices. We combine LoRA-based parameter adaptation with ICL-based conditioning on support examples to enable multi-target prediction. Moreover, other PEFT techniques can be applied under the same interface (Xie et al., 2023; Liu et al., 2024; Luo et al., 2023).

3 Proposed Method

3.1 Problem Formulation

We consider a multi-target prediction problem with N support samples, where each input $x_i \in \mathbb{R}^{n_x}$ is associated with m targets. Here, n_x denotes the number of input features. Let $X = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times n_x}$ denote the input feature matrix, and let $Y_i = (y_i^1, \ldots, y_i^m)$ denote the multi-target vector. The support set is defined as

$$\mathcal{D}^{\mathrm{mt}} = \{(x_i, Y_i)\}_{i=1}^N. \tag{1}$$

Given a query input x^*, our goal is to predict all m targets using both the input features and the observed support targets.

$$P(Y^* \mid x^*, \mathcal{D}^{\mathrm{mt}}). \tag{2}$$

We build on a pretrained single-target TFM f_θ, which takes a query input and a labeled support set $\mathcal{D}^{\mathrm{st}} = \{(x_i, y_i)\}_{i=1}^N$ to approximate the predictive distribution:

$$p_\theta(y^* \mid x^*, \mathcal{D}^{\mathrm{st}}) = f_\theta(x^*, \mathcal{D}^{\mathrm{st}}). \tag{3}$$

The model consists of an input encoder ϕ_x, a target encoder ϕ_y, transformer blocks F_θ, and an output decoder d_θ. Additional notation is summarized in Appendix Sec. B. Our goal is to extend this single-target interface to predict all m targets in one forward pass, allowing each prediction to use observed support information across targets while reusing the pretrained model components.

3.2 Direct Joint Distribution Modeling in Pretraining

A natural approach to exploiting target dependencies is to model the joint predictive distribution during pretraining. However, extending a single-target TFM in this way requires a multi-target pretraining design, and explicit joint-output parameterizations can become increasingly costly as the

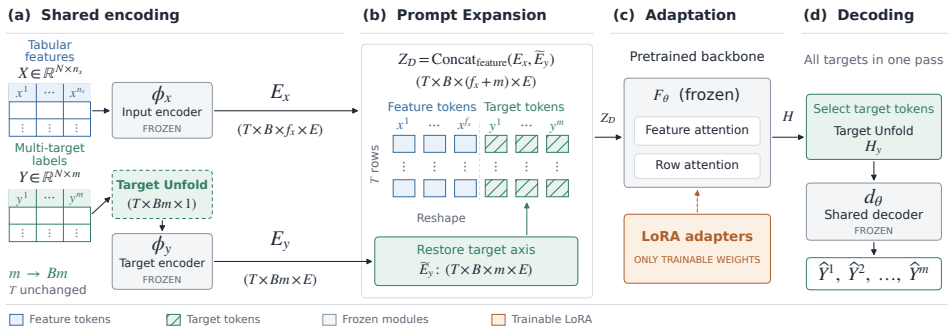

Figure 3: Overview of MT-Tab. The pretrained input encoder ϕ_x, target encoder ϕ_y, output decoder d_θ, and transformer blocks F_θ are frozen. T counts support and query rows, and B is the table/meta-batch size. Only the LoRA weights are activated during training.

number of targets m grows. These costs depend on the distributional assumptions. Appendix Sec. C analyzes this issue for classification and regression and also compares it with native multi-target pretraining approaches such as TabPFN-MT. We instead reuse a pretrained single-target checkpoint through target unfolding and prompt expansion, avoiding additional multi-target pretraining.

3.3 MT-Tab Framework

MT-Tab enables multi-target prediction through interacting target representations and factorized target-wise outputs, without explicitly parameterizing a full joint predictive distribution. Each target is encoded separately using the pretrained single-target encoder through target unfolding, jointly processed with other targets within an expanded transformer prompt, and decoded separately using the shared pretrained decoder. Fig. 3 illustrates this procedure. This architecture-preserving approach incorporates multi-target correlations at inference time through model-specific adaptations, without modifying the core architectures of the underlying TFMs.

Input encoder ϕ_x (frozen). Let T denote the total number of support and query rows, B the table batch size, f_x is the number of encoded input feature tokens per row, and E the embedding dimension. For the tensor operations below, X and Y denote Tabular features and multi-target labels, respectively. The input X is passed through the pretrained input encoder once and shared across all targets:

$$E_x = \phi_x(X) \in \mathbb{R}^{T\times B\times f_x\times E}. \tag{4}$$

Target unfolding and encoding. To reuse the frozen single-target encoder ϕ_y, we unfold the target axis into the batch axis, encode each target, and restore the target axis:

$$\begin{aligned} Y' &= \mathrm{Unfold}(Y) \in \mathbb{R}^{T\times(Bm)\times 1}, \\ \tilde{E}_y &= \mathrm{Reshape}\big(\phi_y(Y')\big) \in \mathbb{R}^{T\times B\times m\times E}. \end{aligned} \tag{5}$$

This preserves a separate embedding for each target while retaining the pretrained single-target encoding interface.

Prompt expansion. We concatenate E_x and the restored $\tilde{E}_y$ along the feature dimension (*prompt expansion*) to obtain the expanded representation, extending the single-target feature axis from f_x+1 to f_x+m:

$$Z_{\mathcal{D}} \;=\; \mathrm{Concat}_{\text{feature}}\big(E_x, \tilde{E}_y\big) \;\in\; \mathbb{R}^{T\times B\times(f_x+m)\times E}. \tag{6}$$

The transformer jointly processes the expanded prompt:

$$H = F_\theta(Z_{\mathcal{D}}) \in \mathbb{R}^{T\times B\times(f_x+m)\times E}, \tag{7}$$

where θ denotes frozen pretrained parameters. Feature-wise attention operates over the f_x+m tokens, enabling explicit interactions among target representations, while item-wise attention integrates information across rows. Together, these operations allow observed support targets to inform the contextualized representations used for query prediction. To adapt the expanded representation Z_D to the pretrained single-target TFM backbone, we learn weights through LoRA.

Shared decoder d_θ (frozen). To feed transformer outputs into the single-target decoder, we apply the same target unfolding. The pretrained decoder d_θ is shared across all targets:

$$\hat{y}^k_{b,j} = d_\theta\big(H_{j,b,k,:}\big), \quad j = 1, \ldots, T,\ b = 1, \ldots, B,\ k = 1, \ldots, m. \tag{8}$$

Multi-target forward. At inference time, MT-Tab enables simultaneous multi-target prediction without explicitly parameterizing a full joint predictive distribution. Target unfolding preserves a separate embedding for each target, while prompt expansion allows the transformer to jointly process feature and target tokens. Attention over the $f_x + m$ tokens enables interactions among target representations before the shared decoder produces target-wise predictions. All targets are predicted in a single forward pass, without conditioning on previously predicted query targets.

Inference cost. MT-Tab shares input encoding and predicts all targets in one forward pass. For feature-wise attention, independent per-target execution requires $\mathcal{O}(mT(f_x + 1)^2)$, compared with $\mathcal{O}(T(f_x + m)^2)$ for MT-Tab. Comparing these cost terms, MT-Tab is less costly when $m < f_x^2$, a condition typically satisfied when the number of targets is modest relative to the number of input feature tokens. Overall runtime is evaluated empirically in Sec. 5.5, and Appendix Sec. G.1. Further details are provided in the appendix, fine-tuning objectives (Sec. D.2), the training algorithm (Alg. 1), and a backbone comparison (Sec. D.3).

4 Experimental Results

4.1 MT-TabBench

We evaluate the multi-target predictive performance of diverse models using *MT-TabBench*, a multi-target benchmark curated from existing single-target datasets and the MultiTab benchmark. A summary of the benchmarked datasets is given in Appendix Table A.9.

OpenML Multi-target benchmark. The OpenML multi-target tabular benchmark includes only datasets from OpenML (Bischl et al., 2017; Gijsbers et al., 2024; Fischer et al., 2023) and TALENT (Liu et al., 2025) that can be restructured in a multi-target format. We restructured the datasets in two ways.

- **`cache_rebuild`**. We identify the unused targets in the original OpenML data and reconfigure them into multi-target datasets.
- **`x2y_conversion`**. We reconfigure certain input features as additional targets, leveraging the inherent multi-task nature of the underlying data.

All datasets are split into $80\%/20\%$ train/test sets. Appendix Table A.10 summarizes the benchmark construction details.

MultiTab benchmark (Sinodinos et al., 2026). We use **ACS Income** and **Higgs** from this large-scale multi-task tabular benchmark. These datasets enable evaluation on large-scale multi-target tabular data and facilitate comparison with MultiTab-Net. We further compare performance across benchmark construction types using group-wise Elo rankings (Appendix Sec. H.3).

4.2 Baselines

We compare MT-Tab with 16 baseline configurations spanning three groups: conventional baselines (MLP and GBDTs), five TFMs evaluated under both prompt evaluation and LoRA, and two supervised multi-target deep models (MultiTab-Net and Mambular). All TFM baselines predict each target independently, requiring m separate inference passes for m targets. Baseline descriptions are provided in Appendix Sec. J.

4.3 Experimental Details

Training details. All training and evaluation runs are repeated five times with different seeds, and we report the mean and standard deviation. Full training details are provided in Appendix Sec. K.

Table 1: Performance on 9 datasets: regression RMSE(↓) and classification AUC(%, ↑), macro-averaged over targets of the same type. **bold** denotes 1st rank and underlined denotes 2nd rank. † denotes partially multi-target models that support multi-target regression but use independent single-target models for classification. All results are measured over five runs.

Method	ACS	Higgs		Energy	SARCOS	Student	Customer	Insurance	Online	OkCupid
	AUC↑	RMSE↓	AUC↑	RMSE↓	RMSE↓	RMSE↓	AUC↑	AUC↑	AUC↑	AUC↑
Single-target models										
LightGBM	89.13 ±0.01	0.275 ±0.00	72.65 ±0.03	0.683 ±0.01	1.085 ±0.00	2.829 ±0.00	93.91 ±0.02	71.29 ±0.14	82.63 ±0.22	62.34 ±1.18
TabPFN-2.5 (Prompt)	87.83 ±0.01	0.274 ±0.01	71.03 ±0.06	0.835 ±0.12	0.604 ±0.00	2.625 ±0.00	94.12 ±0.04	72.76 ±0.12	89.12 ±0.33	72.22 ±0.83
TabPFN-2.5 (LoRA)	87.83 ±0.08	0.265 ±0.00	70.48 ±0.28	0.721 ±0.38	0.599 ±0.00	2.732 ±0.07	94.03 ±0.02	71.94 ±0.06	88.11 ±1.29	71.96 ±3.05
TabICLv2 (Prompt)	88.89 ±0.02	0.278 ±0.00	73.64 ±0.18	0.994 ±0.00	0.556 ±0.00	**2.611** ±0.00	94.72 ±0.02	72.86 ±0.15	89.73 ±0.03	73.30 ±0.27
TabICLv2 (LoRA)	89.00 ±0.02	0.249 ±0.00	75.10 ±0.07	0.447 ±0.06	**0.531** ±0.00	2.627 ±0.01	94.84 ±0.04	72.96 ±0.07	86.48 ±0.25	70.19 ±0.14
TabPFN-3 (Prompt)	88.68 ±0.01	0.259 ±0.00	75.01 ±0.02	0.963 ±0.01	0.607 ±0.01	2.629 ±0.02	94.15 ±0.06	72.54 ±0.29	89.57 ±0.06	72.63 ±0.93
TabPFN-3 (LoRA)	88.19 ±0.05	0.266 ±0.00	72.49 ±0.34	0.623 ±0.24	0.570 ±0.00	2.649 ±0.04	94.49 ±0.07	72.54 ±0.47	89.38 ±0.12	72.32 ±0.90
Exaone-Tabular (Prompt)	88.95 ±0.01	0.261 ±0.00	73.83 ±0.00	0.831 ±0.00	0.616 ±0.00	2.629 ±0.00	94.65 ±0.03	73.36 ±0.02	88.77 ±0.06	74.04 ±0.03
Exaone-Tabular (LoRA)	88.46 ±0.08	0.272 ±0.00	70.65 ±0.14	0.413 ±0.03	0.565 ±0.00	2.637 ±0.01	95.01 ±0.02	73.29 ±0.08	86.95 ±0.06	72.06 ±0.47
TabFM (Prompt)	89.18 ±0.01	0.267 ±0.00	75.21 ±0.01	0.398 ±0.00	0.592 ±0.00	2.706 ±0.00	95.20 ±0.01	**73.58** ±0.00	88.94 ±0.09	**75.70** ±0.04
TabFM (LoRA)	88.97 ±0.00	0.267 ±0.00	74.33 ±0.01	0.365 ±0.00	0.565 ±0.00	2.626 ±0.00	**95.33** ±0.03	73.22 ±0.16	86.23 ±0.02	73.73 ±0.17
Multi-target models										
MLP	83.76 ±1.25	0.296 ±0.00	74.11 ±0.09	0.719 ±0.05	0.953 ±0.00	3.002 ±0.04	93.59 ±0.04	71.51 ±0.13	81.73 ±0.21	67.35 ±0.57
XGBoost†	89.12 ±0.00	0.287 ±0.00	72.98 ±0.05	0.507 ±0.02	1.210 ±0.00	2.763 ±0.05	93.91 ±0.02	71.21 ±0.03	83.66 ±0.09	62.48 ±0.15
CatBoost†	89.31 ±0.01	0.292 ±0.00	72.08 ±0.01	0.526 ±0.01	1.431 ±0.00	2.659 ±0.01	93.78 ±0.01	72.64 ±0.10	85.92 ±0.04	73.02 ±0.56
MultiTab-Net	**89.40** ±0.01	0.233 ±0.00	86.09 ±0.34	2.502 ±0.27	0.943 ±0.02	2.711 ±0.00	94.15 ±0.11	72.60 ±0.27	84.54 ±0.24	72.92 ±0.60
Mambular	86.94 ±0.15	0.337 ±0.00	62.80 ±0.91	2.354 ±0.34	1.407 ±0.02	2.718 ±0.07	92.97 ±0.20	72.42 ±0.22	**90.89** ±1.02	72.04 ±0.53
MT-Tab	88.54 ±0.02	**0.229** ±0.00	**87.72** ±0.08	**0.305** ±0.02	0.618 ±0.00	2.613 ±0.01	95.07 ±0.05	73.08 ±0.08	88.95 ±0.43	74.21 ±0.15

Evaluation Metrics. For each target, we use the following metric according to the task type: Root Mean Squared Error (RMSE) for regression, and area under the ROC curve (AUC) for classification. We report the macro average over targets with the same task type.

MT-Tab backbones. MT-Tab is applicable across multiple TFM backbones (Table A.3). Our implementations follow two integration schemes: concatenation preserves one token per target in TabPFN-2.5 and Exaone-Tabular, whereas summation aggregates target embeddings in TabICLv2 and TabPFN-3. The latter requires adaptations to prompt expansion and decoding, detailed in the appendix Sec. D.3. We use TabPFN-2.5 as the primary backbone because it preserves a distinct token for each target, allowing attention to *explicitly* model interactions among target representations.

4.4 Main Results

Overall performance of MT-Tab. In Table 1, MT-Tab ranks first on 3 of 10 metrics and second on 2 metrics. In Elo rating, MT-Tab is first at 1687.0 and outperforms all baselines (see Fig. 1). It also outperforms GBDT-based methods on all tasks except ACS. In particular, relative to its TabPFN-2.5 backbone, MT-Tab substantially improves Higgs AUC from $71.03 \rightarrow 87.72$ and RMSE from $0.274 \rightarrow 0.229$. Beyond Elo rating, MT-Tab also ranks first under additional evaluation metrics, including competitiveness, average rank, and normalized regret (Appendix Table A.14).

Comparison with single-target TFM LoRA. We compare MT-Tab with applying LoRA to single-target TFMs and running inference independently for each target. On Higgs, MT-Tab improves RMSE from $0.265 \rightarrow 0.229$ and AUC from $70.48 \rightarrow 87.72$, and it also improves RMSE on Energy ($0.721 \rightarrow 0.305$). Its advantage over TFMs-LoRA, which uses the same adaptation strategy, indicates that target unfolding and prompt expansion are effective.

Comparison with Mambular and MultiTab-Net. As shown in Table 1, among non-GBDT multi-target models, MultiTab-Net is strong on ACS AUC (89.40) and Higgs RMSE (0.233), and Mambular ranks first on Online AUC (90.89). However, they degrade substantially on small-scale regression tasks (Energy RMSE 2.502, 2.354). This suggests that MT-Tab has stronger adaptation ability under limited data.

Table 2: Relative performance gap by target correlation. The $\overline{|\rho_S|}$ range denotes the target-correlation values of each benchmark, and Relative gain is the relative performance gap between MT-Tab and TabPFN-2.5 LoRA (RMSE and AUC). All experiments were repeated 5 times.

Benchmark	$\overline{\lvert\rho_S\rvert}$ range	group	MT-Tab	TabPFN-2.5 LoRA	Relative gain (mean)
Energy, Student, SARCOS	$[0.974, 0.349]$	high	1.179	1.351	$+0.196$
ACS, Customer, Higgs	$[0.274, 0.198]$	mid	0.229/90.44	0.265/84.11	$+0.100$
Insurance, Online, OkCupid	$[0.070, 0.031]$	low	78.75	77.34	$+0.019$

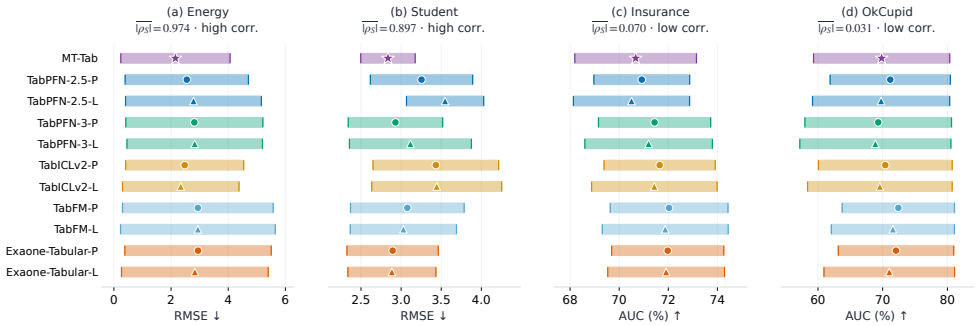

Figure 4: Target-imbalance comparison. Each line spans the sparse-target mean (1% labels) and the dense-target mean (100% labels); end ticks mark those two means and the marker sits at their average. High-correlation Energy and Student are RMSE (lower is better) and low-correlation Insurance and OkCupid are AUC (higher is better). "P" denotes prompt evaluation and "L" denotes LoRA. All results are measured over five runs.

5 Analysis

5.1 Target dependency analysis

To verify that the gains of MT-Tab indeed come from target dependency, we compute, for each benchmark, the mean absolute Spearman correlation $\overline{|\rho_S|}$ over all target pairs and analyze its relationship to the relative performance gain over single-target LoRA with the same backbone.

The higher correlation, the larger average gains. Table 2 computes the mean inter-target correlation for each benchmark and groups them into high, mid, and low according to the range. To measure MT-Tab's target-dependency gain across groups, we compare performance with a single-target LoRA model of the same backbone. As a result, MT-Tab obtains a performance gain of 0.196 over the same single-target model in the High group, 0.1 in the Mid group, and 0.019 in the Low group, empirically confirming that the mean relative gain is larger where target correlations are higher, consistent with the interpretation that the source of the gain is information sharing among targets. Details of the correlation and relative performance-gap calculations and numerical values are described in Appendix Table. A.4.

Additional analyses. Attention to other targets is stronger on highly correlated Student than on weakly correlated OkCupid (Appendix Fig. A.1), while the relative gain on Energy disappears after permuting the auxiliary target (Appendix Table A.5). Both observations are consistent with the correlation–gain trend; details are provided in Appendix Sec. E.

5.2 Target imbalance analysis

We retain all training labels for one dense target and approximately 1% for the remaining sparse targets, with the test split unchanged (Appendix Sec. H.5).

Sparse-target gains under high-correlation benchmarks. On high-correlation Energy and Student, MT-Tab achieves lower sparse-target RMSE (right endpoints) than the strongest single-target baselines (Fig. 4(a), (b)). On low-correlation Insurance and OkCupid, sparse-target AUC (left endpoints) is

Table 3: NPE-PFN comparison: (a) multi-target performance; (b) Energy cooling load RMSE conditioned on predicted ($\hat{y}_h$) or true (y_h) heating load. Results are mean $\pm$ standard deviation over five seeds.

(a) Performance comparison

Dataset (metric)	Independent	NPE-PFN	MT-Tab
Energy (RMSE↓)	0.997 ± 0.036	1.038 ± 0.020	$\mathbf{0.305 \pm 0.020}$
Insurance (AUROC↑)	71.24 ± 0.08	71.95 ± 0.22	$\mathbf{73.08 \pm 0.08}$

(b) Error-propagation analysis

cooling_load query	RMSE↓
$p(y_c \mid x)$	1.548 ± 0.046
$p(y_c \mid x, \hat{y}_h)$	1.673 ± 0.062
$p(y_c \mid x, y_h)$	$\mathbf{1.117 \pm 0.012}$

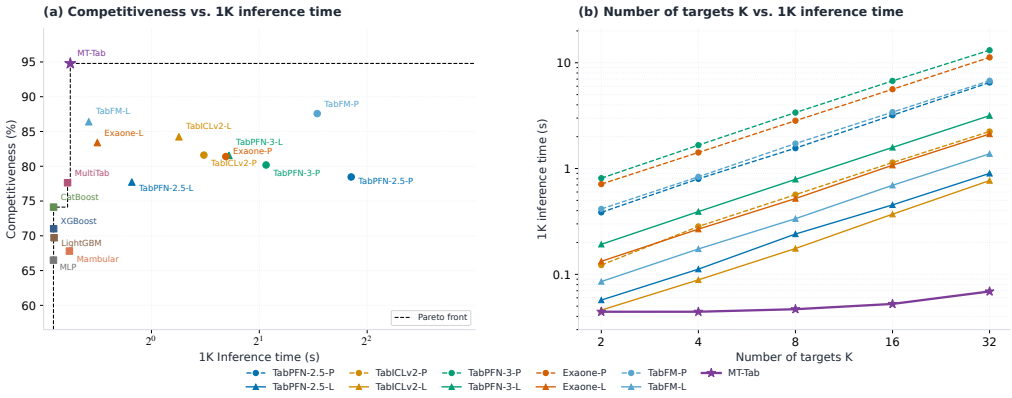

Figure 5: Inference cost analysis. (a) Average competitiveness versus 1K inference time across all datasets. The dashed line denotes the Pareto frontier. (b) 1K inference time as the number of targets K increases. "P" denotes prompt evaluation and "L" denotes LoRA.

nearly identical to same-backbone TabPFN-2.5-LoRA (Fig. 4(c), (d)). Thus, across these four benchmarks, larger sparse-target gains are observed on datasets with higher inter-target correlation.

5.3 Comparison with a sequential conditional model

We compare independent inference and MT-Tab with NPE-PFN (Vetter et al., 2026), which sequentially conditions a frozen TFM on earlier target predictions, using its official code.

Comparison with NPE-PFN. NPE-PFN improves over independent inference on Insurance but degrades on Energy, while MT-Tab performs best on both datasets (Table 3(a)).

Error propagation. For cooling load in Energy, conditioning on predicted heating load increases RMSE from 1.548 to 1.673, whereas conditioning on its true value reduces RMSE to 1.117 (Table 3(b)). This contrast suggests that errors in earlier predictions can offset the benefit of target dependency. MT-Tab jointly uses observed support targets without feeding query-target predictions into subsequent predictions, avoiding this specific propagation path. Related results for GBDT chains (Read et al., 2011; Spyromitros-Xioufis et al., 2016) appear in Appendix Sec. F.

5.4 5-Shot analysis

We further evaluate whether extending a pretrained single-target TFM to multi-target prediction retains its effectiveness under limited supervision. Under the 5-shot protocol, MT-Tab outperforms MLP, MultiTab-Net, and Mambular on all six evaluated tasks (Fig. 2), while remaining competitive with single-target TFMs. Appendix Sec. G provides the full results and detailed comparisons.

5.5 Inference cost

We quantify the computational efficiency gained by the multi-target inference structure of MT-Tab. Competitiveness is computed from benchmark performance, and 1K inference time is measured

Table 4: Ablation of target unfolding, prompt expansion, and the adapter on `Energy` (RMSE↓) and `Insurance` (AUC (%)↑). Checkmarks mark enabled design elements. All results are measured over five runs.

Variant	Target unfolding	Prompt expansion	Adapter	Energy RMSE↓	Insurance AUC (%)↑
TabPFN-2.5-Prompt (Single-target)	–	–	–	0.835 ± 0.124	72.759 ± 0.122
TabPFN-2.5-LoRA (Single-target)	–	–	LoRA	0.721 ± 0.380	71.937 ± 0.059
MT-Tab V1 (Multi-target)	✓	✓	–	1.074 ± 0.000	70.858 ± 0.000
MT-Tab V2 (Multi-target)	Projection layer	✓	–	1.216 ± 0.019	67.115 ± 0.534
MT-Tab V3 (Multi-target)	✓	✓	Full FT	0.880 ± 0.375	71.907 ± 0.270
MT-Tab V4 (Multi-target)	✓	✓	DiffFit	0.436 ± 0.035	73.001 ± 0.107
MT-Tab V5 (Multi-target)	✓	✓	DoRA	0.312 ± 0.022	73.030 ± 0.071
MT-Tab V6 (Multi-target)	✓	✓	RepAdapter	0.284 ± 0.010	73.113 ± 0.222
MT-Tab (Multi-target)	✓	✓	LoRA	0.305 ± 0.015	73.080 ± 0.084

by running inference on 1K benchmark samples. Appendix Sec. I.2 describes the competitiveness calculation. Results are shown in Fig. 5.

Comparison of 1K inference time relative to performance. In Fig. 5(a), MT-Tab achieves higher competitiveness than all baselines. MLP and GBDT families have very short inference times but substantially lower competitiveness. The Pareto frontier consists of the four points {MLP, CatBoost, MultiTab-Net, MT-Tab}, and MT-Tab is the only Pareto-optimal point in the high-competitiveness region. Moreover, MT-Tab has lower 1K inference time than any other single-target TFMs.

1K inference time as the number of targets increases. Single-target inference models require K forward passes for m targets, so inference time grows linearly with m. In contrast, MT-Tab predicts all targets in a *single forward pass*, resulting in nearly constant inference time. In Fig. 5(b), MT-Tab's time increases only $1.56\times$ from $K=2$ to $K=32$, whereas single-target prompt inference increases $16.9\times$. As a result, at $K=32$ it is about $11.2\times$ faster than the fastest single-target TFM, TabICLv2 (LoRA).

5.6 Ablation study

We conduct an ablation study to isolate the contributions of the three core components of MT-Tab: (i) target unfolding, (ii) prompt expansion, and (iii) the adapter used to accommodate the expanded transformer input $(f_x + m)$. Table 4 summarizes the enabled components and adapter choice for each variant, with evaluations on `Energy` and `Insurance`.

V1–V3: Target encoding and backbone adaptation. On Energy and Insurance (Table 4), target unfolding and prompt expansion without backbone adaptation (V1) underperform the single-target prompt baseline; replacing unfolding with a learnable projection layer (V2) further reduces performance. Full fine-tuning (V3) improves upon V1 but remains below the default LoRA variant, indicating that updating all backbone parameters is not necessary for effective adaptation.

V4–V6: Comparison of PEFT adapters. With unfolding and prompt expansion fixed, DiffFit, DoRA, and RepAdapter (V4–V6) achieve broadly comparable performance to LoRA, with RepAdapter yielding the lowest Energy RMSE. We nevertheless retain LoRA as the default choice due to its simplicity and parameter efficiency. Overall, these results show that the proposed framework is not specific to LoRA and remains effective across different parameter-efficient adaptation strategies.

6 Conclusion

We presented MT-Tab, a novel framework that repurposes pretrained single-target tabular foundation models for multi-target prediction through prompt expansion. On the newly constructed benchmark, MT-Tab achieved the highest Elo rating among 17 methods including 16 baselines. Our analyses confirm that MT-Tab effectively leverages pretrained priors to capture target dependencies, delivering significant gains in both predictive accuracy and inference efficiency. These findings suggest that MT-Tab is broadly applicable to various single-target TFMs and serves as a highly scalable framework for multi-target tabular prediction in real-world applications.

AI USE STATEMENT

In this work, we used generative AI tools primarily for language polishing of the draft, including correcting grammar, refining wording, and improving clarity and readability. The authors reviewed and revised the AI-assisted text to ensure that it accurately reflects the methods, results, and intended claims of this work. Generative AI tools were not used to develop the methodology, design or conduct experiments, or analyze the results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

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. *arXiv preprint arXiv:1708.03731*, 2017.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 35(6):7499–7519, 2022.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pp. 227–236. Springer, 1990.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Cormac Cureton and Narges Armanfard. Tabpfn-mt: A natively multitask in-context learner for tabular data. *arXiv preprint arXiv:2605.20234*, 2026.

Thomas G Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286, 1994.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. Exaone tabular 1.0: Technical report. *arXiv preprint arXiv:2608.25774*, 2026.

Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. Openml-ctr23–a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023.

Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data. *arXiv preprint arXiv:2507.03971*, 2025.

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

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520, 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.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

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.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226): 1–16, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Gen Luo, Minglang Huang, Yiyi Zhou, Xiaoshuai Sun, Guannan Jiang, Zhiyu Wang, and Rongrong Ji. Towards efficient visual adaption via structural re-parameterization. *arXiv preprint arXiv:2302.08106*, 2023.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1930–1939, 2018.

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Jingang Qu, David HolzmÃžller, GaÃ⊤l 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 HolzmÃžller, GaÃ⊤l Varoquaux, and Marine Le Morvan. Tabiclv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

Google Research. Tabfm: A zero-shot foundation model for tabular data. 2026. URL `https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/`.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information fusion*, 81:84–90, 2022.

Dimitrios Sinodinos, Jack Yi Wei, and Narges Armanfard. Multitab: A scalable foundation for multitask learning on tabular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25499–25507, 2026.

Eleftherios Spyromitros-Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis Vlahavas. Multi-target regression via input space expansion: treating targets as inputs. *Machine Learning*, 104(1):55–98, 2016.

Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. 2011.

Anton Frederik Thielmann, Manish Kumar, Christoph Weisser, Arik Reuter, Benjamin Säfken, and Soheila Samiee. Mambular: A sequential model for tabular deep learning. *arXiv preprint arXiv:2408.06291*, 2024.

Julius Vetter, Manuel Gloeckler, Daniel Gedon, and Jakob H Macke. Effortless, simulation-efficient bayesian inference using tabular foundation models. *Advances in Neural Information Processing Systems*, 38:82685–82724, 2026.

Enze Xie, Lewei Yao, Han Shi, Zhili Liu, Daquan Zhou, Zhaoqiang Liu, Jiawei Li, and Zhenguo Li. Difffit: Unlocking transferability of large diffusion models via simple parameter-efficient fine-tuning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4207–4216. IEEE, 2023.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609, 2021.

A LIMITATION

Although MT-Tab achieves strong performance and efficiency in tabular multi-target prediction, it has the following limitations.

Dependence on the pretrained backbone. MT-Tab uses a pretrained TFM and trains only LoRA adapters. Therefore, its final performance inherently depends on the quality of the prior learned by the backbone (TabPFN-2.5, TabICLv2, TabPFN-3, Exaone-Tabular, and TabFM). In multi-task settings, because TFMs separate classification and regression tasks, MT-Tab also handles each task type separately. Future work should investigate optimized task-specific backbone selection or unified backbones.

Cost increase with the number of targets and support-set size. MT-Tab unfolds m multi-target labels along the batch dimension and expands the transformer's sequence length to $f_x + m\, f_y$. This design keeps inference time nearly flat with respect to K, but the attention memory cost still increases with sequence length. Therefore, as in existing TFM families, inference cost can grow when the number of targets m or the support-set size N is very large.

ECOC for high-cardinality classification. To utilize the pretrained decoders of TabPFN and TabICL, we introduce Error-Correcting Output Codes (ECOC) when the number of classes C_k satisfies $C_{lim} \leq C_k \leq C_{max}$. ECOC is effective for reusing pretrained priors, but its performance can vary with the codebook design. For multi-target classification problems with very large class counts ($C_k > C_{max}$), we treat them as regression tasks; exceptional cases that require classification remain future work.

B NOTATION SUMMARY

Table A.1: Notation. Tensors are written in row-major order $(T, B, \dots)$, matching Sec. 3.3.

Symbol	Meaning
N, m	number of samples and number of targets
x_j, y_j	input features and single-target label of the j-th sample
X	full input dataset
Y_i	multi-target label
$\mathcal{D}^{\mathrm{mt}}$, $\mathcal{D}^{\mathrm{st}}$	multi-target support set and single-target support set
$\mathcal{D}, \tilde{\mathcal{D}}$	support set and MT-Tab multi-target support set
ϕ_x, ϕ_y	input encoder and target encoder
F_θ	pretrained transformer backbone
d_θ	target decoder
T	number of table rows, counting support and query rows
B	table (meta) batch size; $B = 1$ for single-table adaptation
E	embedding size
n_x	number of input features before encoding
f_x, f_y	input feature tokens and target feature tokens
$\mathrm{TargetUnfold}(\cdot)$	target unfolding: folds the target axis m into the batch axis, $(T, B, m) \to (T, Bm, 1)$
E_x, E_y	encoded feature representation and encoded target representation
$\tilde{E}_y$	target embedding after restoring the target axis, (T, B, m, E)
$Z_{\mathcal{D}}$, H	augmented row representation and transformer output
τ_q	q-th quantile level of the pinball loss
C_k	number of classes for the k-th classification target
r, α	LoRA rank and scaling, $\Delta W = (\alpha/r)\, \Delta W_{\text{low-rank}}$
d_{in}, d_{out}	layer input and output dimensions
$\overline{\lvert\rho_S\rvert}$	mean absolute Spearman correlation over all target pairs of a dataset

C Multi-target Modeling in TFM Pretraining

C.1 Direct Joint Distribution Modelding

Joint predictive distribution. Let $\mathbf{y}_j^* = (y_j^{*,1}, \ldots, y_j^{*,m})$ denote the target vector for query input x_j^*, corresponding to the j-th row of the query target matrix Y^*. A joint predictive distribution can represent dependencies among these targets. By the chain rule of probability (Goodfellow et al., 2016),

$$p(\mathbf{y}_j^* \mid x_j^*, \mathcal{D}^{\mathrm{mt}}) = p(y_j^{*,1} \mid x_j^*, \mathcal{D}^{\mathrm{mt}}) \prod_{k=2}^{m} p\Big(y_j^{*,k} \mid y_j^{*,1:k-1}, x_j^*, \mathcal{D}^{\mathrm{mt}}\Big). \tag{9}$$

The cost of representing the joint distribution depends on its parameterization, as explained below.

Classification with an explicit joint label space. Let $\mathcal{Y}_k$ be the label space of target k, with $C_k = |\mathcal{Y}_k|$ classes. An unrestricted categorical distribution that explicitly enumerates all target combinations has

$$|\mathcal{Y}_1 \times \cdots \times \mathcal{Y}_m| = \prod_{k=1}^{m} C_k \tag{10}$$

possible outcomes. When every target has C classes, this becomes C^m. A dense output head with one logit per joint outcome therefore has an output dimension that grows exponentially with m, $\mathcal{O}(C^m)$.

Regression with a full covariance matrix. For regression targets, consider a multivariate Gaussian predictive distribution:

$$p(\mathbf{y}_j^* \mid x_j^*, \mathcal{D}^{\mathrm{mt}}) = \mathcal{N}(\mathbf{y}_j^*; \mu_j, \Sigma_j), \qquad \mu_j \in \mathbb{R}^m, \quad \Sigma_j \in \mathbb{R}^{m \times m}. \tag{11}$$

Here, μ_j and Σ_j denote the predictive mean vector and covariance matrix conditioned on x_j^* and $\mathcal{D}^{\mathrm{mt}}$, respectively. An unrestricted symmetric covariance matrix contains $m(m+1)/2$ distinct entries, yielding $\mathcal{O}(m^2)$ covariance parameters in addition to the m mean parameters. Thus, explicitly representing pairwise dependencies through a full covariance matrix increases the output parameterization quadratically with the number of targets.

Multi-target Pretraining versus Adaptation. The examples above cover specific joint-distribution parameterizations rather than all multi-target modeling approaches. Methods such as TabPFN-MT (Cureton & Armanfard, 2026) can avoid explicit joint-label enumeration through target-wise outputs based on shared representations, enabling more compact output parameterizations during multi-target pretraining. In contrast, MT-Tab performs downstream adaptation of an existing single-target checkpoint without additional pretraining. Rather than explicitly parameterizing a joint target distribution, it learns shared representations that incorporate cross-target information through target unfolding, prompt expansion, and LoRA adaptation. Appendix Sec. C.2 provides a detailed comparison with TabPFN-MT.

C.2 Comparison with TabPFN-MT

Both TabPFN-MT (Cureton & Armanfard, 2026) and MT-Tab enable simultaneous multi-target prediction, but differ in how they acquire this capability and represent targets within the model.

Pretraining and adaptation cost. TabPFN-MT requires synthetic multi-target prior generation and pretraining, but no downstream weight updates. In contrast, MT-Tab reuses a pretrained single-target checkpoint with downstream LoRA adaptation, avoiding additional multi-target pretraining. Table A.2 reports these distinct training budgets.

Architectural differences. TabPFN-MT projects each sample's target vector into a single embedding and produces target-wise outputs through a shared MLP decoder, whose hidden layers support cross-task information sharing. Its transformer does not explicitly attend between separate target tokens. In contrast, MT-Tab encodes targets separately through target unfolding and combines them with feature embeddings through prompt expansion. Attention over the resulting $f_x + m$ tokens enables explicit cross-target representation interaction within the transformer. The contextualized target representations are then unfolded and decoded separately using the shared single-target decoder.

Scope of comparison. Our comparison focuses on reported training budgets and methodology. The official TabPFN-MT code and pretrained model are not yet available for evaluation, and its evaluation

Table A.2: Reported pretraining and measured adaptation budgets. GPU-hours are computed as GPU count multiplied by elapsed time. Pretraining and downstream adaptation are distinct stages; hardware, model scale, and workload differ across rows.

Model	Stage	Hardware	Elapsed time	GPU-hours
TabPFN-1 (Hollmann et al., 2022)	Pretraining	8× RTX 2080 Ti	20 h	160
TabPFN-2 (Hollmann et al., 2025)	Pretraining	8× RTX 2080 Ti	∼14 days	∼2,688
TabICLv1 (Qu et al., 2025)	Pretraining	3× A100 40GB	20 days	1,440
TabICLv2 (Qu et al., 2026)	Pretraining	4× H100 80GB	∼6.1 days	588
TabPFN-MT (Cureton & Armanfard, 2026)	Multi-target pretraining	8× NVIDIA A5000	∼4 h	∼32
MT-Tab	Downstream LoRA adaptation	1× H100 96GB	≤1 h (max.)	≤1

subsamples cannot be exactly reproduced from the available materials. A controlled performance comparison therefore requires access to the model and matched evaluation subsets. The budgets in Table A.2 describe distinct training stages, rather than establish an end-to-end computational advantage.

D MT-TAB DETAILS

D.1 ALGORITHM

Alg. 1 represents multi-target fine-tuning procedure of MT-Tab from pretrained TFM.

Algorithm 1 Multi-target Fine-tuning of Pretrained TFM (MT-Tab)

Require: Train set $\mathcal{D} = \{(X, Y)\}$, pretrained TFM $f_\theta = (\phi_x, \phi_y, F_\theta, d_\theta)$, epochs $\mathcal{E}$
Ensure: Trained LoRA parameters
Select backbone: $f_\theta \leftarrow$ TabPFN or TabICL
Freeze $\phi_x, \phi_y, F_\theta, d_\theta$
Inject LoRA into item- and feature-attention of F_θ
for epoch $e = 1, \ldots, \mathcal{E}$ **do**
  Split $\mathcal{D} = \mathcal{D}_{\text{prompt}} \cup \mathcal{D}_{\text{holdout}}$
  for each holdout mini-batch $\mathcal{B}_h$ **do**
    $X_{\text{full}} \leftarrow [X_{\text{prompt}}; X_h]$, $Y_{\text{prompt}} \leftarrow Y$ on $\mathcal{D}_{\text{prompt}}$
    $E_x \leftarrow \phi_x(X_{\text{full}})$
    $E_y \leftarrow \phi_y\big(\text{Unfold}(Y_{\text{prompt}})\big)$ ▷ target unfolding
    $Z \leftarrow \text{Concat}_{\text{feature}}(E_x, E_y)$ ▷ prompt expansion
    $H \leftarrow F_\theta(Z; LoRA)$ ▷ LoRA-adapted forward
    $\hat{Y}_k \leftarrow d_\theta(H_{y,k})$ for $k = 1, \ldots, m$ ▷ shared head
    $\mathcal{L} \leftarrow \sum_k M_k \cdot \text{Loss}_k(\hat{Y}_k, Y_{h,k})$
    Backprop and update $\{LoRA\}$ only
  end for
end for

D.2 FINE-TUNING OBJECTIVE

Regression: pinball loss (Steinwart & Christmann, 2011) with quantile unification. We apply the softmax function Bridle (1990) to the output $\hat{y}_{j,k}$ and map it to Q quantiles:

$$\hat{q}_q \;=\; \text{QuantileTransform}\big(\text{softmax}(\hat{y}_{j,k})\big), \qquad \tau_q = \frac{q + 0.5}{Q}, \;\; q = 0, \ldots, Q-1, \tag{12}$$

$$\mathcal{L}_{\text{pin}}^{(k)} \;=\; \frac{1}{Q} \sum_{q=0}^{Q-1} \max\big(\tau_q \, e_{q,k}, \; (\tau_q - 1)\, e_{q,k}\big), \qquad e_{q,k} = y_k - \hat{q}_{q,k}. \tag{13}$$

We train regression tasks with this pinball loss. The point estimate is obtained by averaging the quantiles.

Classification: cross-entropy loss (Goodfellow et al., 2016). For classification tasks, we apply the softmax function to the output $\hat{y}_{j,k}$ and train with cross-entropy loss:

$$\mathcal{L}_{\text{cls}}^{(k)} = \text{CE}\big(\text{softmax}(\hat{y}_{j,k,1:c_k}), y_k\big). \tag{14}$$

ECOC. TabPFN and TabICL apply ECOC (Dietterich & Bakiri, 1994) when the class count C_k exceeds $C_{lim} > 10$. Similarly, for MT-Tab classification, when a target has $C_{lim} \leq C_k \leq C_{max}$ classes, we use ECOC to utilize the pretrained decoder. We set $C_{max} = 25$. For a codebook $\{b^{(r)}\}_{r=1}^{R}$, where $b^{(r)} : \{1, \ldots, C_k\} \to \{0, 1\}$,

$$\mathcal{L}_{\text{ECOC}}^{(k)} = \frac{1}{R} \sum_{r=1}^{R} \text{BCE}\big(\hat{y}_k^{(r)}, b^{(r)}(y_k)\big), \tag{15}$$

where BCE means binary cross-entropy function. At inference time, the class is recovered from the R binary outputs through minimum-distance decoding.

Total loss. We apply task weights w_{reg}, w_{cls} and train with the loss assigned to each task.

$$\mathcal{L} = \frac{1}{\sum_{j,k}} \sum_{j,k} \Big(w_{\text{reg}}\, \mathcal{L}_{\text{reg},j}^{(k)} + w_{\text{cls}}\, \mathcal{L}_{\text{cls},j}^{(k)} \Big). \tag{16}$$

D.3 MT-Tab (Other TFM Backbones)

We extend MT-Tab to TabICLv2, TabPFN-3, and Exaone-Tabular. MT-Tab supports both concatenation-based (TabPFN-2.5 and Exaone-Tabular) and summation-based (TabICLv2 and TabPFN-3) integration of target embeddings. We adapt prompt expansion and decoding according to the integration scheme. For the backbone comparison in Table A.3, we select Energy (regression) and Customer (classification) for their relatively high mean absolute target correlations ($\overline{|\rho_S|} = 0.974$ and 0.266, respectively; Table A.4).

Table A.3: Backbone comparison on `Energy` (RMSE↓) and `Customer` (AUC (%)↑). Under each dataset we report mean±std for Prompt (single-target), LoRA (single-target), and MT-Tab (multi-target).

	Energy (RMSE↓)			**Customer** (AUC (%)↑)		
Backbone	Prompt (Single)	LoRA (Single)	MT-Tab (Multi)	Prompt (Single)	LoRA (Single)	MT-Tab (Multi)
TabICLv2	0.994 ±0.00	0.447 ±0.06	0.414 ±0.01 ▲0.033	**94.72** ±0.02	94.84 ±0.04	95.05 ±0.04 ▲0.21
TabPFN-3	0.963 ±0.01	0.623 ±0.24	0.368 ±0.04 ▲0.255	94.15 ±0.06	94.49 ±0.07	94.93 ±0.05 ▲0.44
Exaone-Tabular	**0.831** ±0.00	**0.413** ±0.03	0.392 ±0.04 ▲0.021	94.65 ±0.03	**95.01** ±0.02	95.03 ±0.02 ▲0.02
TabPFN-2.5	0.835 ±0.12	0.721 ±0.38	**0.305** ±0.02 ▲0.416	94.12 ±0.04	94.03 ±0.02	**95.07** ±0.05 ▲1.04

TabICLv2 and TabPFN-3: prompt expansion via element-wise summation. Unlike the concatenation-based interfaces of TabPFN-2.5 and Exaone-Tabular, TabICLv2 and TabPFN-3 backbone expects a single embedding aligned along the token axis. Replacing $\text{Concat}_{\text{feature}}$ in Eq. 6, we collapse the m-axis of $\tilde{E}_y$ via summation and combine it with E_x through element-wise summation ($\oplus$):

$$Z_{\mathcal{D}} = E_x \oplus \sum_{k=1}^{m} \tilde{E}_y^{(k)} \in \mathbb{R}^{T \times B \times E}. \tag{17}$$

TabICLv2 and TabPFN-3 cannot disentangle which target each transformer output corresponds to. Therefore, in MT-Tab (TabICLv2 and TabPFN-3), we replace the shared decoder of Eq. 8 with m per-target copies; the base weights are kept frozen, and only target-specific adaptation modules are trained.

Backbone comparison. Table A.3 shows that MT-Tab improves mean performance over the corresponding single-target LoRA baselines for all four evaluated backbones on both Energy and Customer,

supporting its applicability across different backbone architectures. In our implementations, TabPFN-2.5 and Exaone-Tabular preserve a distinct token for each target through concatenation, enabling explicit attention between target representations. TabICLv2 and TabPFN-3 instead aggregate target embeddings through summation, integrating multi-target information without maintaining separate target tokens within the transformer. We use TabPFN-2.5 as the default backbone because its target-token-preserving structure enables explicit cross-target attention.

E TARGET CORRELATION ANALYSIS

Correlation measurement. We compute correlations in Table A.4, which summarizes the number of target pairs, $\overline{|\rho_S|}$ (mean absolute Spearman correlation), $\max|\rho_S|$, mean residual Pearson, mean NMI, and mean Cramér's V for each dataset. Datasets are sorted in descending order of $\max|\rho_S|$.

Table A.4: Summary of target dependencies in benchmark datasets. $\overline{|\rho_S|}$ and $\max|\rho_S|$ denote the mean and maximum absolute Spearman correlation over all target pairs. $\overline{r_{\text{resid}}}$ denotes residual Pearson, and $\overline{\text{NMI}}$ means Normalized Mutual Information.

Dataset	n_y	Pairs	$\overline{\lvert\rho_S\rvert}$	Max $\lvert\rho_S\rvert$	residual	NMI
Energy	2	1	0.974	0.974	0.801	0.551
Student	3	3	0.897	0.938	n/a	0.416
SARCOS	7	21	0.349	0.964	0.219	0.097
ACS	2	1	0.274	0.274	n/a	0.048
Customer	4	6	0.266	0.523	n/a	0.086
Higgs	8	28	0.198	0.870	0.266	0.034
Insurance	4	6	0.070	0.160	n/a	0.007
Online	3	3	0.052	0.100	n/a	0.015
OkCupid	3	3	0.031	0.040	n/a	0.005

Mean correlation index $\overline{|\rho_S|}$. If a dataset has n_y targets, the number of pairs of distinct targets is $n_y(n_y-1)/2$. The Pairs column of Table A.4 reports this count. For each pair we compute the Spearman rank correlation ρ_S, take the absolute value, and average those values arithmetically to obtain $\overline{|\rho_S|}$. Same-target pairs are excluded, and the quantity is the mean over all pairs rather than the single strongest pair. Thus it measures, on average, how strong the linear correlation among targets is for that benchmark.

$$\overline{|\rho_S|} \;=\; \frac{2}{n_y(n_y-1)} \sum_{1\le i<j\le n_y} \big|\rho_S(y_i,y_j)\big|. \tag{18}$$

Relative gain. The relative performance gap compares single-target LoRA and MT-Tab with the same backbone, isolating the comparison as much as possible from LoRA itself.

$$\text{gain}_{\text{rel}} = \begin{cases} (\,\text{st}-\text{mt}\,)/|\text{st}| & \text{(RMSE)} \\ (\,\text{mt}-\text{st}\,)/|\text{st}| & \text{(AUC)} \end{cases} \tag{19}$$

Here mt and st are the scores of MT-Tab and single-target LoRA from Table 1, respectively; a positive value means MT-Tab is better.

Target correlation and attention. To examine how attention varies with target correlation, we compare Student ($\overline{|\rho_S|}$=0.897) and OkCupid (0.031), both with three targets. Fig. A.1 shows the target–target block of feature-attention weights from trained MT-Tab checkpoints, averaged over all layers and heads. We subtract the uniform baseline $1/s$ to account for the different token counts, counting packed input-feature tokens rather than raw input columns, where $s = f_x + m$ (the number of input feature tokens and targets, respectively). On Student, all off-diagonal entries exceed the uniform baseline, with a mean excess of $+0.0240$, and are comparable to self-attention. On OkCupid, all off-diagonal entries fall below the baseline, with a mean excess of -0.0128, and self-attention is strongest in each row. Thus, attention to other targets is more prominent on the highly correlated benchmark, consistent with the larger gains observed at higher target correlations in Sec. 5.1.

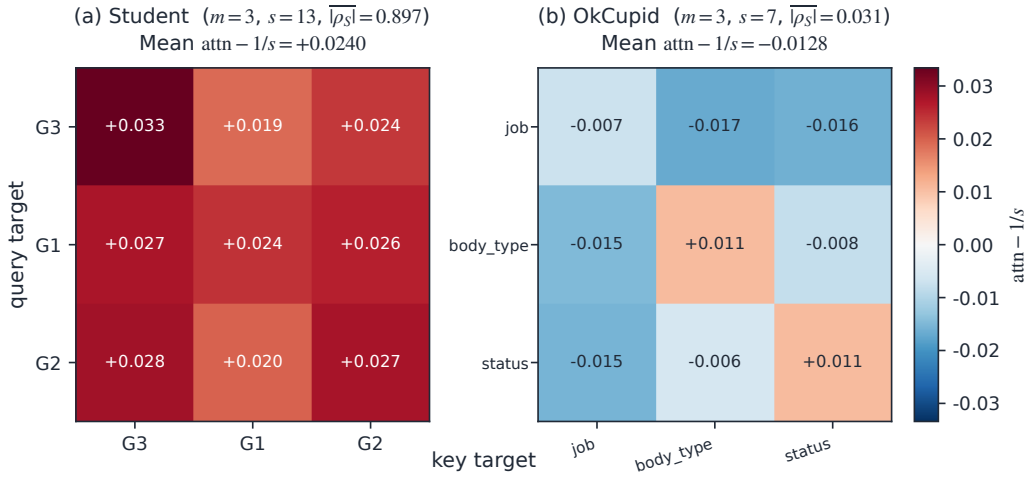

Figure A.1: Target–target attention map of MT-Tab, averaged over all layers and heads, for (a) Student and (b) OkCupid. Colors show $\mathrm{attn} - 1/s$ on a shared scale centered at zero, where $s = f_x + m$. Positive values indicate attention above the uniform baseline.

Table A.5: Target-permutation analysis on Energy. Original scores are from Table 1; permuted scores are averaged over five seeds. RMSE is macro-averaged over both targets. Relative gain follows Eq. 19; positive values favor MT-Tab.

Variant	$\overline{\lvert\rho_S\rvert}$	MT-Tab RMSE↓	TabPFN-2.5 LoRA RMSE↓	Relative gain↑
Original (`Energy`)	0.974	0.305 ± 0.02	0.721 ± 0.38	$+0.577$
Permuted (`Energy_p`)	0.010	5.173 ± 0.08	5.134 ± 0.06	-0.008

Target permutation on Energy. To complement the comparisons across benchmarks, we examine whether the gain of MT-Tab persists when target dependence is disrupted within Energy. We keep "*heating_load*" and the input features unchanged and randomly permute "*cooling_load*" within each train and test split, yielding `Energy_p`. This preserves dataset size and target marginals. Apart from the permutation, all experimental conditions remain unchanged, including the data splits, input features, TabPFN-2.5 backbone, and each model's LoRA protocol from Table 1.

Target permutation eliminates the relative gain of MT-Tab. Table A.5 shows that $\overline{|\rho_S|}$ drops from 0.974 to 0.010, while the relative gain of MT-Tab falls from $+0.577$ to -0.008. This result is consistent with the correlation–gain trend in Sec. 5.1. However, permutation also disrupts the relationship between cooling load and the inputs, increasing both models' RMSE. We therefore compare relative gains within each variant.

F GBDT-Chain

Table A.6: Performance comparison of GBDT models with classifier chains and regressor chains against MT-Tab on the 5-shot multi-target benchmark. All results are measured over five runs.

Method	Classification (AUC %, ↑)			Regression (RMSE, ↓)			Rank↓
	ACS	Insurance	OkCupid	Student	Higgs (R)	Energy	AVG.
LightGBM	50.00 ±0.00	63.86 ±0.02	56.36 ±0.08	3.152 ±0.00	0.413 ±0.00	**4.176** ±0.04	4.75
LightGBM (chains)	50.00 ±0.00	63.59 ±0.05	57.76 ±0.21	3.152 ±0.00	0.413 ±0.00	9.986 ±0.00	5.42
XGBoost	61.01 ±0.53	66.56 ±0.37	53.92 ±0.13	3.199 ±0.01	0.420 ±0.01	**4.176** ±0.05	5.00
XGBoost (chains)	65.43 ±0.32	66.50 ±0.08	56.87 ±0.10	3.290 ±0.03	0.483 ±0.00	4.415 ±0.12	5.33
CatBoost	61.38 ±1.81	66.87 ±0.20	60.51 ±1.92	3.002 ±0.01	0.395 ±0.00	**4.176** ±0.06	2.17
CatBoost (chains)	61.19 ±0.31	66.67 ±0.13	59.03 ±0.21	3.182 ±0.07	0.399 ±0.00	4.396 ±0.14	3.83
MT-Tab	**69.96** ±0.30	**68.37** ±0.07	**60.83** ±0.83	**2.952** ±0.05	**0.376** ±0.00	4.244 ±0.01	**1.50**

Classifier & Regressor chains. We also evaluated classifier chains (Read et al., 2011) and regressor chains (Spyromitros-Xioufis et al., 2016) in the 5-shot benchmark, enabling GBDT models to exploit target dependencies. Since standard classifier chains are designed for binary multi-target prediction, they are not directly applicable to MT-TabBench, which includes multi-class targets. Therefore, we implement a multiclass classifier chains variant and include it in our experiments, and we use regression chains without modification. Table A.6 shows chain-based GBDTs generally underperform the vanilla GBDTs and MT-Tab. This is also reflected in the Rank column. These results indicate that in few-shot settings, chain-based GBDTs suffer from error propagation, leading to unstable performance. In contrast, MT-Tab has more stability in multi-target problems.

G Additional 5-Shot analysis

Evaluation setup. We complement the analysis in Sec. 5.4 with detailed results on six tasks under limited supervision. We follow the 5-shot sampling protocol described in Appendix Sec. H.4: for classification, we sample five rows per class for each target; for regression, we sample five rows per target. The final training set is the union of the sampled rows across targets, and the original test split is preserved. We report the mean and standard deviation over five runs.

Table A.7: 5-shot multi-target benchmark performance. **bold** denotes 1st rank and underlined denotes 2nd rank. † denotes partially multi-target models that support multi-target regression but use independent single-target models for classification.

Method	Classification (AUC %, ↑)			Regression (RMSE, ↓)			Rank↓
	ACS	Insurance	OkCupid	Student	Higgs (R)	Energy	AVG.
Single-target models							
LightGBM	50.00 ±0.00	63.86 ±0.02	56.36 ±0.08	3.152 ±0.00	0.413 ±0.00	4.176 ±0.04	12.83
TabPFN-2.5 (Prompt)	64.22 ±0.27	**68.83** ±0.14	62.75 ±0.85	2.942 ±0.00	0.393 ±0.00	4.162 ±0.03	5.00
TabPFN-2.5 (LoRA)	63.33 ±0.50	68.59 ±0.57	63.11 ±1.09	**2.940** ±0.00	0.383 ±0.00	4.070 ±0.08	4.50
TabICLv2 (Prompt)	69.33 ±0.39	68.56 ±0.06	60.70 ±0.36	3.308 ±0.05	0.468 ±0.00	3.837 ±0.07	8.33
TabICLv2 (LoRA)	68.36 ±0.03	68.64 ±0.01	59.27 ±0.01	3.356 ±0.00	0.463 ±0.00	**3.809** ±0.00	8.67
TabPFN-3 (Prompt)	62.37 ±0.48	68.80 ±0.56	63.16 ±1.45	2.950 ±0.01	0.381 ±0.00	4.371 ±0.11	5.67
TabPFN-3 (LoRA)	57.86 ±1.63	67.18 ±0.67	58.82 ±0.54	2.963 ±0.10	0.418 ±0.01	3.947 ±0.13	9.33
Exaone-Tabular (Prompt)	**74.29** ±0.00	68.76 ±0.00	**63.62** ±0.07	2.960 ±0.00	0.380 ±0.00	4.150 ±0.00	**3.33**
Exaone-Tabular (LoRA)	58.06 ±1.16	66.93 ±0.07	57.52 ±1.23	3.181 ±0.14	0.408 ±0.02	4.364 ±0.22	12.00
TabFM (Prompt)	68.14 ±0.00	66.35 ±0.00	59.75 ±0.09	3.061 ±0.00	0.388 ±0.00	3.936 ±0.00	7.33
TabFM (LoRA)	58.94 ±1.89	58.62 ±0.58	59.38 ±0.71	2.951 ±0.02	0.386 ±0.00	4.101 ±0.10	8.83
Multi-target models							
MLP	65.72 ±0.70	48.56 ±0.66	58.09 ±0.94	3.130 ±0.01	0.439 ±0.00	5.992 ±0.46	12.67
XGBoost†	61.01 ±0.53	66.56 ±0.37	53.92 ±0.13	3.199 ±0.01	0.420 ±0.00	4.176 ±0.04	12.67
CatBoost†	61.38 ±1.81	66.87 ±0.20	60.51 ±1.92	3.002 ±0.01	0.395 ±0.00	4.176 ±0.06	9.00
MultiTab-Net	54.44 ±3.49	62.65 ±1.17	53.32 ±2.94	3.174 ±0.10	0.435 ±0.03	10.100 ±0.24	15.17
Mambular	65.03 ±4.02	65.93 ±0.60	56.30 ±1.96	3.052 ±0.05	0.438 ±0.01	6.071 ±0.70	12.33
MT-Tab	69.96 ±0.30	68.37 ±0.07	60.83 ±0.83	2.952 ±0.05	**0.376** ±0.00	4.244 ±0.01	5.33

Comparison with supervised multi-target baselines. Table A.7 shows that MT-Tab outperforms MLP, MultiTab-Net, and Mambular on all six tasks, consistent with Fig. 2. It also outperforms the evaluated GBDT baselines on five of the six tasks, with Energy being the exception. These results support the effectiveness of adapting a pretrained TFM for multi-target prediction when labeled data are scarce.

Comparison with single-target TFMs. MT-Tab remains competitive with single-target TFMs, achieving the lowest Higgs RMSE (0.376) and the second-highest ACS AUC (69.96%) among all evaluated methods. Its average rank is 5.33, compared with 5.00 for TabPFN-2.5 (Prompt) and 4.50 for TabPFN-2.5 (LoRA), which use the same backbone. Relative to both same-backbone baselines, MT-Tab performs better on ACS and Higgs but worse on the remaining four tasks. Thus, the 5-shot results support the viability of the multi-target extension under limited supervision, while also showing that it does not uniformly improve upon independent single-target prediction.

G.1 TIME-MEMORY TRADE-OFF IN SINGLE-TARGET TFMS

To measure both the wall-time reduction that single-target TFMs can obtain via batch-forward and the accompanying memory increase, we measured wall time and GPU peak memory for $m \in \{8, 16, 32\}$ with the same backbone.

Table A.8: Independent sequential/batched execution versus shared execution. Time is the mean and sample standard deviation over 5 outer repeats.

K	Execution mode	Time (s)↓	Peak GPU memory (bytes)↓	Memory ratio vs. MT-Tab
8	Independent batched	0.00284 ± 0.000091	77244928	1.911
8	Independent sequential	0.02161 ± 0.000866	40089088	—
8	MT-Tab	0.00281 ± 0.000067	40412672	—
16	Independent batched	0.00319 ± 0.000076	119697408	2.934
16	Independent sequential	0.04384 ± 0.001050	40095232	—
16	MT-Tab	0.00281 ± 0.000083	40790016	—
32	Independent batched	0.00429 ± 0.000069	204602368	4.925
32	Independent sequential	0.09117 ± 0.003292	40107520	—
32	MT-Tab	0.00290 ± 0.000063	41544704	—

The wall time of sequential grows approximately with m. The wall time of batched is approximately $7.6\times$, $13.7\times$, and $21.3\times$ faster than that of the serial case, and it grows moderately with m. The wall time of MT-Tab remains nearly constant across the full range. For GPU peak memory, the peak of batched grows with m, whereas the peaks of MT-Tab and sequential remain nearly constant at approximately 38–40 MB. That is, single-target TFMs can reduce inference time via batching at the cost of higher memory consumption, while reducing memory usage requires sequential execution and thus increases inference time. This results in a time–memory trade-off for single-target TFMs, whereas MT-Tab remains efficient in both time and memory.

H MT-TABBENCH

We summarize the sources of the benchmarks used in our experiments and the reproduction procedure. MT-TabBench consists of two benchmark suites: (i) the **OpenML Multi-target** benchmark newly curated from original OpenML datasets (eight datasets), and (ii) the public multi-task benchmark **MultiTab** (ACS Income and Higgs).

Table A.9: Original benchmark splits and targets: Table 1 excluding Auction. n_x and n_y count input and target columns. The last column counts unique target values, not classes for regression.

Dataset	Task	Train rows	Test rows	n_x	n_y	Unique values per target
Energy	Regression	614	154	8	2	heating: 587; cooling: 636
SARCOS	Regression	39,146	9,787	21	7	V22–V28: 11,414; 9,635; 8,738; 7,907; 5,985; 5,998; 5,893
Student	Regression	519	130	30	3	G3: 17; G1: 17; G2: 16
ACS	Classification	1,165,150	249,675	10	2	MAR: 5; PINCP: 2
Customer	Classification	103,904	25,976	18	4	satisfaction: 2; Class: 3; CustomerType: 2; TypeofTravel: 2
Insurance	Classification	18,838	4,710	7	4	Response: 2; Accomodation_Type: 2; Holding_Policy_Type: 4; Reco_Insurance_Type: 2
Online	Classification	9,864	2,466	15	3	Revenue: 2; TrafficType: 21; VisitorType: 3
OkCupid	Classification	21,341	5,336	11	3	job: 3; body_type: 12; status: 5
Higgs	Regression 7 + classification 1	9,900,000	550,000	21	8	Target: 2; m_{bb}: 1.12M; m_{jj}: 1.06M; m_{jjj}: 487K; m_{jlv}: 517K; m_{lv}: 341K; m_{wbb}: 684K; m_{wwbb}: 762K

H.1 OPENML MULTI-TARGET BENCHMARK

This benchmark includes OpenML and TALENT datasets that can be represented in a multi-target format. The reproduction method is summarized in Sec. 4.1 and Table A.10.

Motivation. Multi-target prediction arises in settings where *multiple outcomes must be judged jointly from a single observation*, such as multiple quality indicators in manufacturing, multiple risk metrics in finance, and co-occurring diagnoses in healthcare. However, public tabular benchmarks are mostly normalized to a single target, so no standard suite exists for evaluating multi-target methods. MT-TabBench fills this gap by restructuring public datasets into a multi-target format. Benchmark

selection used the criteria of (i) public availability, (ii) presence of relationships among targets, (iii) diversity of task types, and (iv) grounding in application context.

Reproducibility. To ensure reproducibility, we release the benchmark and the generation code for the OpenML multi-target benchmark.

Table A.10: Reproduction information for datasets included in the OpenML multi-target benchmark.

Dataset	Build kind	Task ID	Target columns (m)
energy_efficiency	cache_rebuild	361617	y_heating_load, y_cooling_load
sarcos	cache_rebuild	361254	y_V22, y_V23, y_V24, y_V25, y_V26, y_V27, y_V28
student_performance_por	cache_rebuild	361619	y_G1, y_G2, y_G3
auction_verification	cache_rebuild	361236	y_verification_time (reg), y_verification_result (cls)
customer_satisfaction_in_airline	x2y_conversion	363628	y_satisfaction, y_Class, y_CustomerType, y_TypeofTravel
insurance	x2y_conversion	45064	y_Response, y_Accomodation_Type, y_Holding_Policy_Type, y_Reco_Insurance_Type
online_shoppers	x2y_conversion	363691	y_Revenue, y_TrafficType, y_VisitorType
okcupid_stem	x2y_conversion	45067	y_job, y_body_type, y_status

H.2 MULTITAB BENCHMARK

MultiTab is a public standard benchmark for multi-task tabular learning. We use the following two datasets without modification.

- **ACS Income**: A multi-target classification. The two targets are `Income` (binary) and `Marital` (multi-class), which are jointly predicted from the same input features.
- **Higgs**: A multi-task dataset. It includes regression targets for seven high-level physics quantities ($m_{jj}, m_{jjj}, m_{lv}, m_{jlv}, m_{bb}, m_{wbb}, m_{wwbb}$) and one binary classification target.

H.3 PERFORMANCE BY BENCHMARK CONSTRUCTION TYPE

Construction-wise evaluation. Following the curation criteria described above, we distinguish Native (ACS, Higgs-R, Higgs-C), Cache rebuild (Energy, SARCOS, Student), and x2y conversion (Customer, Insurance, Online, OkCupid). All includes the metrics from all three groups. We recompute Elo within each group using the same 17 methods and mean scores from Table 1, following Appendix Sec. I.1.

Table A.11: Top-five Elo ranks by benchmark construction type. Cells show methods and Elo scores. P/L denote Prompt/LoRA. N_m counts metrics. Ratings are fitted independently within each group.

Rank	Native ($N_m = 3$)	Cache rebuild ($N_m = 3$)	x2y conversion ($N_m = 4$)	All ($N_m = 10$)
Rank 1	MultiTab-Net (1896.5)	TabFM-L (1758.1)	TabFM-P (1788.0)	MT-Tab (1687.0)
Rank 2	MT-Tab TabICLv2-L (1573.8)	TabICLv2-L (1707.2)	MT-Tab (1674.7)	TabFM-P (1670.9)
Rank 3	—	MT-Tab (1630.7)	TabICLv2-P Exaone-Tabular-P (1600.6)	TabFM-L (1631.4)
Rank 4	TabFM-P (1539.8)	Exaone-Tabular-L (1591.1)	—	TabICLv2-L (1584.8)
Rank 5	TabPFN-3-P (1428.3)	TabICLv2-P (1542.1)	TabFM-L (1583.3)	TabICLv2-P (1536.7)

Distinct group winners. Table A.11 shows that MultiTab-Net ranks first on Native, TabFM (LoRA) on Cache rebuild, and TabFM (Prompt) on x2y conversion. These distinct winners indicate that MT-TabBench captures different strengths across methods and construction settings. Although MT-Tab leads no individual group, it ranks second on Native, third on Cache rebuild, and second on x2y conversion. Including all 10 metrics yields an Elo of 1687.0, placing MT-Tab first overall and highlighting its consistently strong performance across construction types.

H.4 5-SHOT BENCHMARK

The 5-shot benchmark used in Sec. 5.4 is constructed by resampling the same original datasets defined in Sec. H according to the 5-shot rule. The original test split is preserved.

Sampling rule.

- **Classification:** For each target, we sample five rows per class without replacement. In multi-target datasets, the final training set is the union of samples obtained across all targets.
- **Regression:** For each target, we sample five rows and take their union.

The results reported in Table A.7 of Sec. 5.4 are computed on the following six datasets. Table A.12 summarizes the number of training rows, test rows, feature columns, and target-wise sample numbers.

Table A.12: Configuration of datasets used in the 5-shot analysis in Sec. 5.4. Source train denotes the number of rows in the original training data, and 5-shot train denotes the number of rows after resampling. Test rows and #col are inherited from the original test split and feature columns. Target sampling reports the per-target sample numbers.

Dataset	Task type	Source train	5-shot train	Test rows	#col	Target sampling
Classification						
acs_income	multi-target classification	1,165,150	35	249,675	12	MAR: 5×5, PINCP: 2×5
insurance	multi-target classification	18,838	50	4,710	11	Response: 2×5, Accomodation_Type: 2×5, Holding_Policy_Type: 4×5, Reco_Insurance_Type: 2×5
okcupid_stem	multi-target classification	21,341	100	5,336	14	job: 3×5, body_type: 12×5, status: 5×5
Regression						
student_performance_por	multi-target regression	519	15	130	33	G1: 5, G2: 5, G3: 5
higgs (regression)	multi-target regression	9,900,000	35	550,000	28	m_{jj}, m_{jjj}, m_{lv}, m_{jlv}, m_{bb}, m_{Wbb}, m_{Wwbb}: each 5
energy_efficiency	multi-target regression	614	10	154	10	heating_load: 5, cooling_load: 5

H.5 TARGET IMBALANCE BENCHMARK

Benchmark setup. For each dataset in the full multi-target benchmark, we keep one dense target with 100% of labels and retain only 1% of labels for the remaining targets during training. The original test set is kept unchanged. The results in Sec. 5.2 use the two highest-correlation datasets (`energy_efficiency`, `student_performance_por`) and the two lowest-correlation datasets (`insurance`, `okcupid_stem`) from Sec. E. Dense/sparse target configurations and remaining-label counts are summarized in Table A.13.

Table A.13: Configuration of target-imbalance benchmark. The "Dense" target retains 100% of training labels, while "Sparse" targets retain only 1% of labels. The "effective ratio" is the actual remaining label ratio.

Dataset	Source benchmark	train	test	n_x	n_y	Dense (100%)	Sparse (~1%)	eff. ratio
energy_efficiency_imbalance	energy_efficiency	614	154	8	2	y_heating_load	y_cooling_load (6 / 614)	0.977%
student_performance_por_imbalance	student_performance_por	519	130	30	3	y_G1	y_G2 (5/519), y_G3 (5/519)	0.963%
insurance_imbalance	insurance	18,838	4,710	7	4	y_Accomodation_Type	y_Response (188/18838), y_Holding_Policy_Type (188/18838), y_Reco_Insurance_Type (188/18838)	0.998%
okcupid_stem_imbalance	okcupid_stem	21,341	5,336	11	3	y_job	y_body_type (213/21341), y_status (213/21341)	0.998%

I METRICS

I.1 ELO RATING

To compare overall model rankings, we compute an Elo rating based on the Bradley–Terry model Bradley & Terry (1952). For each evaluation metric (RMSE or AUC), we compare every model pair (i, j) and construct a win matrix W_{win}. A model wins if it has a lower value for RMSE or a higher value for AUC.

$$W_{win_{ij}} = \sum_{b=1}^{B_{bm}} \mathbf{1}\big[\text{model } i \text{ beats model } j \text{ on benchmark } b\big], \tag{20}$$

where B_{bm} is the total number of benchmark metrics. For the rating vector $\gamma \in \mathbb{R}^n$, where n is the number of models, we iteratively apply the following update until convergence:

$$\gamma_i \leftarrow \frac{\sum_{j \neq i} W_{win_{ij}}}{\sum_{j \neq i} \frac{W_{win_{ij}} + W_{win_{ji}}}{\gamma_i + \gamma_j}}. \tag{21}$$

After each iteration, γ is normalized to have mean 1. The converged rating is converted to the Elo scale as follows, where 1500 is the baseline rating:

$$\text{Elo}_i = 1500 + 400 \cdot \log_{10}\left(\frac{\gamma_i}{\bar{\gamma}}\right), \tag{22}$$

where $\bar{\gamma}$ is the mean rating across all models. We assume that the performance of each (model, benchmark) pair follows $\mathcal{N}(\mu, \sigma^2)$ and perform 1,000 bootstrap samples, using the mean and standard deviation from repeated experiments as μ and σ^2. We recompute Elo for each sample and report the standard deviation of each model's Elo as uncertainty.

I.2 Competitiveness

Improvability is a metric used by TabICLv2 Qu et al. (2026) that measures how far a method falls behind the best-performing method on each dataset. We define competitiveness as its complementary metric, measuring how close a method is to the best-performing method. For model m on dataset i, competitiveness is defined as follows:

$$\text{err}_i^* = \min_{m'} \text{err}_i(m'), \tag{23}$$

$$\text{Competitiveness}_i(m) = \frac{\text{err}_i^*}{\text{err}_i(m)} \times 100, \tag{24}$$

where err represents RMSE or $1 - \text{AUC}/100$. The final competitiveness score is the average over all datasets:

$$\text{Competitiveness}(m) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \text{Competitiveness}_i(m). \tag{25}$$

I.3 Average Rank

Let $r_b(m)$ be the rank of model m on metric b, with lower RMSE or higher AUC ranked first and exact ties assigned average ranks. We report

$$\text{AvgRank}(m) = \frac{1}{B_{bm}} \sum_{b=1}^{B_{bm}} r_b(m), \tag{26}$$

where lower is better.

I.4 Win/Tie/Loss Counts

W/T/L counts pairwise wins, ties, and losses over $16 \times 10 = 160$ comparisons per model. Lower RMSE or higher AUC wins; exact equality is a tie. Results are ranked by $W + T/2$ (higher is better).

I.5 Normalized Regret

Normalized regret scales the gap to the best model by the error range on each metric:

$$\text{NormRegret}(m) = \frac{1}{B_{bm}} \sum_{b=1}^{B_{bm}} \frac{\text{err}_b(m) - \text{err}_b^*}{\text{err}_b^{\text{worst}} - \text{err}_b^*}. \tag{27}$$

Here $\text{err}_b(m)$ is RMSE or $1 - \text{AUC}/100$, and err_b^* and $\text{err}_b^{\text{worst}}$ are the minimum and maximum errors across models. Values range from 0 (best) to 1 (worst), reflecting relative performance gaps.

I.6 Overall Comparison

Elo rating does not capture performance-gap magnitudes; we therefore also report competitiveness and normalized regret, alongside rank and pairwise summaries. All aggregates use the 17 methods and $B_{bm} = 10$ metrics in Table 1. Table A.14 compares all methods using the five summaries. MT-Tab ranks first under all five summaries, followed by TabFM (Prompt), supporting its leading performance under both ordinal and gap-sensitive metrics.

Table A.14: Overall comparison on the 10 metrics, ordered by Elo. Comp. denotes competitiveness. **Bold**/underlined values indicate best/second-best results; † follows Table 1.

Method	Elo↑	Comp. (%)↑	Avg. rank↓	W/T/L	Norm. regret↓
MT-Tab	**1687.0**	**94.79**	**4.30**	**127/0/33**	**0.090**
TabFM (Prompt)	<u>1670.9</u>	<u>87.57</u>	<u>4.55</u>	<u>124/1/35</u>	<u>0.151</u>
TabFM (LoRA)	1631.4	86.40	5.20	117/2/41	0.188
TabICLv2 (LoRA)	1584.8	84.24	6.00	110/0/50	0.223
TabICLv2 (Prompt)	1536.7	81.61	6.90	101/0/59	0.232
Exaone-Tabular (Prompt)	1534.6	81.40	6.95	100/1/59	0.205
TabPFN-3 (Prompt)	1478.1	80.18	8.05	88/3/69	0.266
Exaone-Tabular (LoRA)	1477.0	83.40	8.05	89/1/70	0.236
MultiTab-Net	1447.9	77.63	8.65	83/1/76	0.363
TabPFN-3 (LoRA)	1447.9	81.57	8.65	83/1/76	0.267
TabPFN-2.5 (Prompt)	1393.7	78.46	9.75	72/1/87	0.304
CatBoost†	1344.4	74.13	10.70	63/0/97	0.425
TabPFN-2.5 (LoRA)	1292.9	77.73	11.65	53/1/106	0.370
XGBoost†	1236.5	71.02	12.65	43/1/116	0.579
LightGBM	1224.1	69.73	12.85	41/1/118	0.589
Mambular	1162.1	67.80	13.80	32/0/128	0.638
MLP	1125.4	66.51	14.30	27/0/133	0.706

J Baseline Descriptions

We consider 16 baseline configurations: four conventional baselines, ten TFM configurations obtained by evaluating five model families under two adaptation regimes, and two supervised multi-target deep models.

ML baselines. We include (i) **MLP**, (ii) **XGBoost** (Chen & Guestrin, 2016), (iii) **CatBoost** (Prokhorenkova et al., 2018), and (iv) **LightGBM** (Ke et al., 2017). MLP inherently supports multi-target outputs. For GBDT-based models, we utilize their native multi-target support for regression; however, for classification, we train independent models for each target as they do not natively support multi-label classification.

TFMs. We evaluate five recent TFM families under two regimes, *Prompt Evaluation* (zero/few-shot inference) and *LoRA Fine-tuning*: (v-vi) **TabPFN-2.5** (Grinsztajn et al., 2025), (vii-viii) **TabICLv2** (Qu et al., 2026), (ix-x) **TabPFN-3** (Grinsztajn et al., 2026), (xi-xii) **Exaone-Tabular** (Eo et al., 2026), (xiii-xiv) **TabFM** (Research, 2026). Note that all TFM baselines follow a single-target interface, thus requiring m independent inference passes for m targets, which neglects inter-target dependencies.

Supervised multi-target deep models. (xv) **MultiTab-Net** (Sinodinos et al., 2026) is a transformer-based model specifically designed for multi-task tabular data. (xvi) **Mambular** (Thielmann et al., 2024) is a Mamba-based tabular deep learning model that processes feature tokens sequentially. Both methods support multi-target prediction in a supervised-learning setting.

Implementation details, hyperparameters, and training settings for all baselines are provided in Appendix Sec. K.

K TRAINING DETAILS

We summarize the hyperparameters of all baselines and MT-Tab. All experiments are conducted on an NVIDIA H100 96GB GPU.

K.1 COMMON SETTINGS

Train/Validation/Test split. We use the officially provided train/test split and reserve 5% of the training set for validation. The best checkpoint is selected according to the average validation metric.

Prompt size (N_{prompt}). For TFM-based methods (LoRA / MT-Tab), the maximum support-set size of training is set according to the following rule for large-scale datasets (`ACS Income`, `Higgs`, `sarcos`, `customer_satisfaction_in_airline`) due to GPU constraints:

- Large-scale datasets: $N_{\text{prompt}} = 30{,}000$.
- Other datasets: $N_{\text{prompt}} = 50{,}000$ (default).

For inference, we use the maximum support-set size of evaluation: 50,000.

- $N_{\text{prompt}} = 50{,}000$.

K.2 MLP

We implement MLP directly in PyTorch (Paszke et al., 2019).

- Layer: 4-layer MLP, hidden width 256, with multi-target head.
- Activation function: ReLU
- dropout = 0.1
- Optimizer: Adam
- Learning rate: 1×10^{-3}
- Epoch: 200, early stopping patience 20.

K.3 GBDT (XGBOOST / CATBOOST / LIGHTGBM)

We evaluate GDBT models using their respective libraries.

- Multi-target handling: automatically selected according to each model's native multi-target support
  - XGBoost/CatBoost use multi-target learning for regression and independent single-target learning for classification.
  - LightGBM uses independent single-target learning.
- Default hyperparameters from each library.
- early stopping: 20
- validation metric: RMSE or logloss, depending on the task.

K.4 MULTITAB-NET

We use the official implementation of MultiTab-Net (Sinodinos et al., 2026)[1]. For the MultiTab benchmark, we use the provided default hyperparameters. For the OpenML multi-target benchmark, the main hyperparameters are as follows:

- Learning rate: 1×10^{-2}
- Optimizer: AdamW (Loshchilov & Hutter, 2017)
- Batch size: 2,048
- Max epochs: 300, early stopping patience 30

[1] https://github.com/Armanfard-Lab/MultiTab

K.5 MAMBULAR

We use the official implementation of MAmbular (Thielmann et al., 2024)[2].

- Learning rate: 1×10^{-4}
- Optimizer: Adam
- Batch size: 64
- Max epochs: 50, early stopping patience 10

K.6 TFM PROMPT EVALUATION

We perform single-target inference, without training the pretrained TFMs. We use the official implementations of both TabPFN-2.5 and TabPFN-3[3], TabICLv2[4], Exaone-Tabular[5], and TabFM[6].

- Pretrained backbone: official checkpoint
- $N_{\text{estimators}} = 8$
- Train Prompt size: the common rule in Sec. K.1
- Evaluation Prompt size = $50k$

K.7 TFM LORA FINE-TUNING

LoRA fine-tuning trains only low-rank adapters on top of a frozen backbone. the additional LoRA-specific settings are as follows:

- Learning rate: 3×10^{-4}
- Optimizer: AdamW
- Epochs: 100
- LoRA rank: $r = 16$
- LoRA alpha: $\alpha = 32$ $(\alpha/r = 2)$
- LoRA dropout: 0.1
- Training Prompt size: large-scale datasets = $30k$, others = $50k$
- Evaluation Prompt size = $50k$
- Loss: selected according to the task.
  - regression: pinball loss
  - classification: cross-entropy loss

K.8 MT-TAB

Because LoRA fine-tuning in MT-Tab jointly adapts prompt expansion and downstream prediction, we train for more epochs.

- Backbone: TabPFN-2.5
- Learning rate: 3×10^{-4}
- Optimizer: AdamW
- Epochs: 100
- LoRA rank: $r = 16$

[2] https://github.com/Ovler-Young/DeepTabular
[3] https://github.com/PriorLabs/TabPFN
[4] https://github.com/soda-inria/tabicl
[5] https://github.com/LGAI-Research/EXAONE-Tabular
[6] https://github.com/google-research/tabfm

- LoRA alpha: $\alpha = 32$
- LoRA dropout: 0.1
- Training Prompt size: large-scale datasets = $30k$, others = $50k$
- Evaluation Prompt size: $50k$
- Loss: selected according to the task.
  - regression: pinball loss
  - classification: cross-entropy loss