MaskTab: Scalable Masked Tabular Pretraining with Scaling Laws and Distillation for Industrial Classification

Anonymous ACL submission

Abstract

Tabular data forms the backbone of high-stakes decision systems in finance, healthcare, and beyond. Yet industrial tabular datasets are inherently difficult: high-dimensional, riddled with missing entries, and rarely labeled at scale. While foundation models have revolutionized vision and language, tabular learning still leans on handcrafted features and lacks a general self-supervised framework. We present MaskTab, a unified pre-training framework designed specifically for industrial-scale tabular data. MaskTab encodes missing values via dedicated learnable tokens, enabling the model to distinguish structural absence from random dropout. It jointly optimizes a hybrid supervised pre-training scheme—utilizing a twin-path architecture to reconcile masked reconstruction with task-specific supervision—and an MoE-augmented loss that adaptively routes features through specialized subnetworks. On industrial-scale benchmarks, it achieves +5.04% AUC and +8.28% KS over prior art under rigorous scaling. Moreover, its representations distill effectively into lightweight models, yielding +2.55% AUC and +4.85% KS under strict latency and interpretability constraints, while improving robustness to distribution shifts. Our work demonstrates that tabular data admits a foundation-model treatment—when its structural idiosyncrasies are respected.

1 Introduction

Tabular data lies at the core of high-stakes automated decision systems—credit scoring, risk prediction, and fraud detection—where model reliability directly impacts financial outcomes. Yet despite its operational centrality, tabular representation learning has seen little of the paradigm shift brought by self-supervised pretraining in other domains. Real-world industrial tables are lagre-sacle, sparse, and only weakly labeled; more critically, missing values are often informative rather than ignorable, reflecting underlying data-generating processes. Standard pipelines, however, treat such structure as noise: features with high missing rates are discarded, continuous variables are naively imputed, and categorical fields are reduced to static encodings. The result is a loss of signal before learning even beigins.

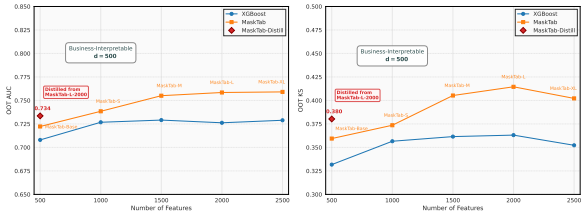

Figure 1: Overall Model Performance. Average monthly AUC (left) and KS (right) on OOT data for MaskTab and XGBoost (Chen and Guestrin, 2016), as the number of features increases from 500 to 2500. MaskTab consistently outperforms XGBoost, with larger gains at higher feature dimensions. `MaskTab-Distill` further transfers the benefits of `MaskTab-L` to a 500-feature, interpretable setting, outperforming both `MaskTab-Base` and XGBoost.

Current practice reflects this gap. Gradient-boosted Decision trees (GBDTs) like XGBoost (Chen and Guestrin, 2016) dominate production pipelines not because they are optimal, but because they tolerate raw inputs with minimal preprocessing. Deep alternatives such as FT-Transformer (Gorishniy et al., 2021) improve expressivity but still treat tabular data as a passive collection of columns, ignoring the semantics of absence and scaling poorly beyond a few hundred features. Crucially, none offer a self-supervised pretraining regime that leverages unlabeled data at scale—a missed opportunity given the abundance of unlabeled records in enterprise settings.

We contend that tabular data can support foundation models, provided the architecture respects

its statistical idiosyncrasies. To this end, we introduce MaskTab, a pretraining framework built on three design principles:(1)*Missingness as signal:* Instead of imputing or dropping incomplete entries, MaskTab represents missing values with dedicated learnable tokens, allowing the model to distinguish informative absence from noise. (2) *Unified pretraining:* A Siamese twin-path architecture jointly optimizes masked reconstruction and supervised objectives, enabling seamless adaptation from pretraining to downstream tasks—even with scarce labels. (3) *Scalable interaction modeling:* A Mixture-of-Experts (MoE) reconstruction head adaptively routes high-dimensional features through specialized subnetworks, capturing complex interactions without combinatorial explosion.

As shown in Figure 1, MaskTab consistently outperforms XGBoost across feature scales (500–2500), with gains widening as dimensionality grows. Notably, a distilled variant—constrained to 500 interpretable features—surpasses both XGBoost and our base model, demonstrating that rich representations can be compressed without sacrificing performance or robustness to temporal shifts.

Our results establish that tabular data admits a foundation-model treatment when representation learning is grounded in its operational reality. MaskTab sets a new state-of-the-art benchmark on industrial datasets, achieving a relative improvement of 5.04% in AUC and 8.28% in KS over existing baselines. Furthermore, to address the stringent latency and interpretability requirements of production environments, we develop a knowledge distillation pipeline. The resulting lightweight student models retain the powerful semantics of the teacher, yielding a +2.55% AUC and +4.85% KS gain over traditional models while exhibiting superior robustness to temporal distribution shifts. More broadly, our results suggest that tabular data should no longer be regarded as a "solved" engineering problem, but as an open frontier for representation learning.

2 Related Work

Tabular Models. In tabular data modeling, tree-based ensembles such as XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) have long been established as strong baselines due to their robustness, interpretability, and performance on structured data. With advancements in deep learning, research has shifted towards deep tabular models like ResNet (Gorishniy et al., 2021), SNN (Klambauer et al., 2017), and DCNv2 (Wang et al., 2020). More recently, Transformer-based approaches have gained prominence by treating individual features as tokens and embedding feature values into high-dimensional spaces. Notable methods in this line include TransTab (Wang and Sun, 2022), FT-Transformer (Gorishniy et al., 2021), which leverage self-attention to capture complex feature interactions. Further studies have proposed improved embedding techniques for different data types, such as enhanced representations for numerical features (Gorishniy et al., 2022) and pre-trained BERT embeddings for categorical/textual attributes (Ye et al., 2024), respectively. Another line of work adopts neighborhood-based retrieval, where test samples are compared against the training set to identify nearest neighbors for prediction, as seen in TabR (Gorishniy et al., 2024).

Scaling Law. Scaling laws characterize the predictable relationship between model performance and key scaling variables such as model size, dataset size, and computational budget. Early work by (Kaplan et al., 2020) established power-law scaling in language models, later refined by DeepMind's Chinchilla laws (Hoffmann et al., 2022) and Gopher study (Rae et al., 2021). Similar trends have been observed in vision (Zhai et al., 2022) and domain-specific settings such as recommender systems (Lai et al., 2025) and time series modeling (Yao et al., 2024). For tabular data, Ma et al. (2024) recently initiated scaling studies, yet most efforts focus on model and data scaling, overlooking feature dimension scaling and unlabeled data utilization—especially under realistic conditions with heterogeneous types and systematic missingness.

Knowledge Distillation. Knowledge distillation (KD) is a widely used technique for compressing large models into smaller, efficient variants while preserving performance. (Hinton et al., 2015) established the paradigm of using softened teacher logits ("soft targets") as training labels for the student, capturing dark knowledge via KL divergence loss. In tabular data, Knowledge Distillation has been used to transfer expertise from dominant tree ensembles like GBDTs into neural networks, as seen in DeepGBM (Ke et al., 2019), which leverages tree outputs to guide the student model. More recently, (Wang and Sun, 2022) applied distillation in their TransTab framework, where a large

transformer-based teacher model pre-trained on diverse tabular datasets transfers knowledge to a smaller student model for improved generalization on downstream tasks. However, distilling tabular models—especially those handling missing values and heterogeneous groups—under strict latency and memory constraints remains under-explored.

3 MaskTab

3.1 Task Definition

We study tabular classification with a labeled set and a much larger unlabeled set:

$$\mathcal{D} = \mathcal{D}_{\text{sup}} \cup \mathcal{D}_{\text{unsup}}, \quad \mathcal{D}_{\text{sup}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}, \quad \mathcal{D}_{\text{unsup}} = \{\mathbf{x}_j\}_{j=1}^{M}, \quad M \gg N. \tag{1}$$

Each instance $\mathbf{x} = (v_1, \ldots, v_d)$ contains heterogeneous features (numeric, categorical, text, or missing). We use binary classification as a running example, with labels $y \in \{0, 1\}$, however, MaskTab naturally extends to n-way classification and regression.

3.2 Baseline: Transformer Encoder for Heterogeneous Tables

We adopt an encoder-only Transformer as the backbone for modeling feature interactions in heterogeneous tabular data. For each feature k, we construct a token embedding by combining a feature-name embedding and a feature-value embedding:

$$\mathbf{h}_k = \text{LayerNorm}\big(\mathbf{e}_k^{\text{name}} + \phi(v_k)\big) \in \mathbb{R}^h. \tag{2}$$

Feature names are encoded once using a frozen pre-trained language encoder (BERT (Devlin et al., 2018)) with pooling to obtain $\mathbf{e}_k^{\text{name}}$. The value encoder $\phi(\cdot)$ is type-specific: a linear projection for numerical values, an embedding lookup for categorical values, a frozen text encoder for textual values, and a shared learnable embedding for missing values.

Stacking all feature tokens yields $\mathbf{H} \in \mathbb{R}^{d \times h}$, which is fed into a Transformer to capture cross-feature dependencies:

$$\mathbf{Z} = \text{Transformer}(\mathbf{H}). \tag{3}$$

We obtain a row-level representation by mean pooling and apply a linear classification head to predict $\hat{y}$. The model is trained on $\mathcal{D}_{\text{sup}}$ with binary cross-entropy loss. Building on this, we introduce MaskTab, as shown in Figure 2.

3.3 Learnable Tokens for Masked and Real Missing Values

We propose a self-supervised objective that treats missingness as a learnable signal and unifies *synthetic masking* and *natural missing values* with dedicated tokens. Concretely, we use two special symbols: `[MASK]` for randomly masked observed entries ($\mathcal{M}$) during pre-training and `[MISS]` for naturally missing entries ($v_k = \perp$). Their embeddings are initialized identically with a shared learnable vector $\mathbf{m} \in \mathbb{R}^h$, referred to as the *mask embedding*, which reduces the mismatch between pre-training and real missingness:

$$\phi(v_k) = \begin{cases} \mathbf{e}_{\texttt{[MASK]}}, & k \in \mathcal{M}, \\ \mathbf{e}_{\texttt{[MISS]}}, & v_k = \perp, \\ \text{standard encoding}, & \text{otherwise}, \end{cases}$$
$$\mathbf{e}_{\texttt{[MASK]}} = \mathbf{e}_{\texttt{[MISS]}} = \mathbf{m} \quad \text{(at init)}. \tag{4}$$

Masked value reconstruction. Given an unlabeled instance $\mathbf{x} = (v_1, \ldots, v_d)$, we sample a subset of observed features $\mathcal{M}$ and replace them with `[MASK]` to obtain $\widetilde{\mathbf{x}}$. After encoding, $\mathbf{Z} = \text{Transformer}_\Theta(\phi(\widetilde{\mathbf{x}}))$, we predict each masked value from its contextual representation:

$$\mathcal{L}_{\text{MLM}} = \sum_{k \in \mathcal{M}} \ell(g(\mathbf{z}_k),\, v_k)\,, \tag{5}$$

where $g(\cdot)$ is a lightweight prediction head shared across features and ℓ is type-aware (e.g., MSE for numerical features and cross-entropy/contrastive loss for categorical or textual features). This objective encourages the encoder to exploit both feature co-occurrence patterns and missingness structure.

3.4 Hybrid Supervised Pre-Training with Twin Networks

Conventional two-stage pre-training (self-supervised pre-train $\rightarrow$ supervised fine-tune) is often suboptimal for industrial tabular learning. It (i) requires separate hyperparameter schedules across stages, and (ii) suffers from *objective misalignment*: the checkpoint minimizing the reconstruction loss does not necessarily optimize downstream classification, even when substantial labeled data are available.

We therefore adopt a *hybrid supervised* objective that trains on labeled and unlabeled data in a single, unified procedure. For labeled samples $(\mathbf{x}, y)$, we jointly optimize masked reconstruction

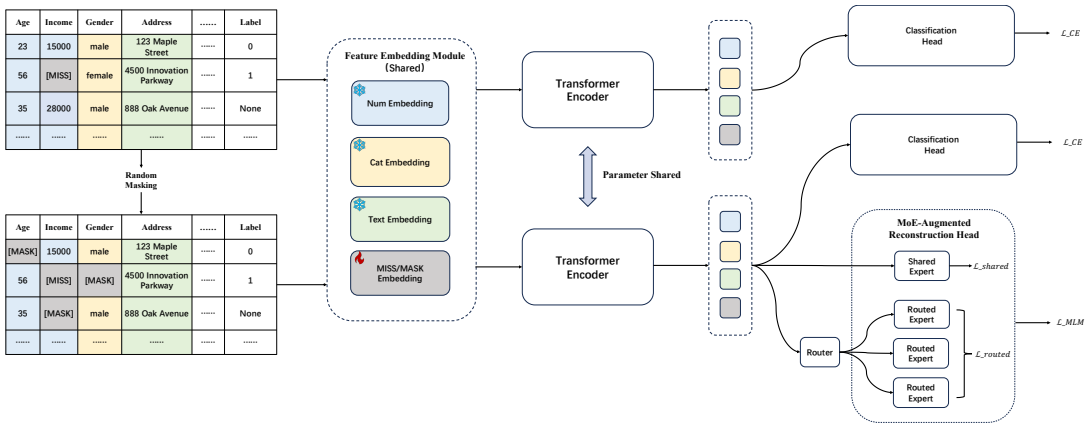

Figure 2: MaskTab encodes high-dimensional tabular data with learnable tokens, then hybrid pre-trains a tabular encoder with a twin-path objective: MoE-based masked reconstruction on unlabeled data and supervised learning on labeled data.

and classification:

$$\mathcal{L}_{\text{hybrid}}^{(\text{sup})} = \lambda\, \mathcal{L}_{\text{MLM}} + (1-\lambda)\, \mathcal{L}_{\text{CE}}, \quad \lambda \in [0,1], \tag{6}$$

where $\mathcal{L}_{\text{MLM}}$ is the reconstruction loss in Sec. 3.3 and $\mathcal{L}_{\text{CE}}$ is the binary cross-entropy loss. For unlabeled samples $\mathbf{x}$, we apply only the self-supervised objective:

$$\mathcal{L}_{\text{hybrid}}^{(\text{self-sup})} = \mathcal{L}_{\text{MLM}}. \tag{7}$$

The overall objective averages over labeled and unlabeled sets:

$$\begin{aligned}\mathcal{L}_{\text{hybrid}} = \frac{1}{|\mathcal{D}_{\text{sup}}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{sup}}} \mathcal{L}_{\text{hybrid}}^{(\text{sup})} \\ + \frac{1}{|\mathcal{D}_{\text{unsup}}|} \sum_{\mathbf{x}\in\mathcal{D}_{\text{unsup}}} \mathcal{L}_{\text{hybrid}}^{(\text{self-sup})}.\end{aligned} \tag{8}$$

We use the same formulation throughout training, avoiding an explicit pre-train/fine-tune boundary.

Twin-path training to avoid masking-induced shift. Directly feeding masked inputs to the classifier trains $P(y \mid \widetilde{\mathbf{x}})$, introducing a train–test mismatch because inference uses the natural missing pattern in $\mathbf{x}$. To eliminate this shift, for each labeled instance we employ two parallel paths with shared Transformer parameters Θ (Fig. 2):

Reconstruction path. We apply masking to obtain $\widetilde{\mathbf{x}}$ and compute

$$\begin{aligned}\mathbf{Z}^{(\text{MLM})} &= \text{Transformer}_\Theta(\phi(\widetilde{\mathbf{x}})), \\ &\mathcal{L}_{\text{CE}}\left(\mathbf{z}^{(\text{MLM})}\right), \qquad \mathcal{L}_{\text{MLM}}(\mathbf{Z}^{(\text{MLM})}).\end{aligned} \tag{9}$$

Classification path. We keep the original input $\mathbf{x}$ and compute

$$\begin{aligned}\mathbf{z}^{(\text{CLS})} &= \text{Pooling}\big(\text{Transformer}_\Theta\big(\phi(\mathbf{x})\big)\big), \\ &\mathcal{L}_{\text{CE}}\left(\mathbf{z}^{(\text{CLS})}\right).\end{aligned} \tag{10}$$

Sharing Θ transfers dependency/missingness knowledge learned by reconstruction to the classifier, while ensuring the classifier is trained on the true input distribution $P(y \mid \mathbf{x})$.

Adaptive masking. To stabilize training for inherently sparse instances, we reduce corruption as the original missing ratio increases. Let $\eta(\mathbf{x}) = \frac{1}{d}\sum_{k=1}^{d} \mathbb{I}(v_k = \perp)$ be the missing ratio and $\eta_{\max}$ the maximum over the dataset. We set the instance-wise masking rate as

$$r_{\text{mask}}(\mathbf{x}) = r_{\max}\left(1 - \frac{\eta(\mathbf{x})}{\eta_{\max}}\right)^{\alpha}, \tag{11}$$

where $r_{\max}$ is the maximal masking ratio and $\alpha > 0$ controls sensitivity. This preserves information for highly-missing samples while still enforcing strong corruption for complete ones.

3.5 Scalable Mixture-of-Experts Feature Grouping

A single linear projection for masked feature reconstruction becomes a bottleneck as the number of tabular features grows. We therefore replace the monolithic MLM head with a lightweight Mixture-of-Experts (MoE) projection layer, inspired by recent MoE LLMs (e.g., DeepSeekMoE; Dai et al., 2024). The key idea is to *route features to specialized experts*: each feature token selects a small subset of experts for reconstruction, increasing capacity while keeping per-token computation bounded.

Concretely, we combine (i) one *shared* expert, applied to all masked features, and (ii) K_r *routed* experts with top-K_a gating. The MLM objective becomes

$$\mathcal{L}_{\text{MLM}} = \alpha\,\mathcal{L}_{\text{Shared}} + \beta\,\mathcal{L}_{\text{Routed}}, \tag{12}$$

where α, β weight the shared and routed terms, and $\mathcal{M}$ denotes masked positions. The shared expert loss is

$$\mathcal{L}_{\text{Shared}} = \sum_{k\in\mathcal{M}} \ell(\mathbf{w}_0\mathbf{z}_k + \mathbf{b}_0,\, v_k)\,, \tag{13}$$

and the routed expert loss is

$$\mathcal{L}_{\text{Routed}} = \sum_{k\in\mathcal{M}} \ell\left(\sum_{i=1}^{K_r} g_{i,k}\big(\mathbf{w}_i\mathbf{z}_k + \mathbf{b}_i\big),\, v_k\right). \tag{14}$$

For feature k, the gate $g_{i,k}$ keeps only the top-K_a routed experts:

$$g_{i,k} = \begin{cases} s_{i,k}, & s_{i,k} \in \text{TopK}(\{s_{j,k}\}_{j=1}^{K_r},\, K_a), \\ 0, & \text{otherwise}, \end{cases} \tag{15}$$

with routing scores computed by centroid matching

$$s_{i,k} = \text{Softmax}_i\left(\mathbf{z}_k^\top \mathbf{e}_i\right), \tag{16}$$

where $\mathbf{z}_k$ is the token representation, $\{\mathbf{e}_i\}_{i=1}^{K_r}$ are learnable expert centroids, and $(\mathbf{w}_i, \mathbf{b}_i)$ are expert-specific output projections. This design encourages adaptive feature grouping and expert specialization, substantially improving reconstruction capacity without introducing heavy expert MLPs.

3.6 Task-Specialized Distillation with Interpretable Features

Despite their formidable capabilities, pre-trained models are often prohibitively large and utilize an extensive set of features, rendering them unsuitable for scenarios like credit scoring that demand low latency and high interpretability. To derive a practical model, we distill its knowledge into a smaller version of MaskTab that uses only a predefined, interpretable feature set.

The student model is trained by employing a embedding alignment loss to match the teacher's output distribution, combined with a hybrid-supervision loss (detailed in Section 3.4). Specifically, to align the intermediate representations, the student's final hidden state $\mathbf{z}^{(\text{CLS})}$ is first projected into the teacher's embedding space using a linear layer that increases its dimensionality to match that of the teacher's representation using a linear layer that increases its dimensionality from d_s to d_t:

$$\mathbf{z}_a = \mathbf{w}_a\mathbf{z}^{(\text{CLS})} + \mathbf{b}_a, \tag{17}$$

where $\mathbf{w}_a \in \mathbb{R}^{d_t\times d_s}$ and $\mathbf{b}_a \in \mathbb{R}^{d_t}$ are learnable parameters. We then minimize a representation loss $\mathcal{L}_{\text{align}}$ defined as the negative cosine similarity between the projected student representation $\mathbf{z}_a$ and the embedding $\mathbf{e}_t \in \mathbb{R}^{d_t}$ generated by the `MaskTab-L` teacher:

$$\mathcal{L}_{\text{align}} = 1 - \cos(\mathbf{z}_a, \mathbf{e}_t) = 1 - \frac{\mathbf{z}_a \cdot \mathbf{e}_t}{|\mathbf{z}_a||\mathbf{e}_t|}. \tag{18}$$

Therefore, the overall training objective for the student model is a weighted combination of multiple losses:

$$\begin{aligned}\mathcal{L}_{\text{hybrid}}^{(\text{sup})} =& \lambda_1 \cdot \mathcal{L}_{\text{MLM}} + \lambda_2 \cdot \mathcal{L}_{\text{CE}} + \lambda_3 \cdot \mathcal{L}_{\text{align}}, \\ & \lambda_1 + \lambda_2 + \lambda_3 = 1.\end{aligned} \tag{19}$$

4 Experiments

4.1 Industrial-scale data setting

We evaluate MaskTab on tabular benchmarks that capture practical industrial constraints. Specifically, we use the public TabReD benchmark (Rubachev et al., 2024), which comprises eight datasets spanning three classification and five regression tasks, and a proprietary CreditRisk dataset containing 6.4×10^5 labeled instances and 1.3×10^7 unlabeled instances collected in 2024. For CreditRisk, we enforce a strictly temporal split to prevent information leakage: labeled data are partitioned by month into Training (Jan–Jun 2024; 330k), Validation (Jul–Sep 2024; 180k), and Test (Oct–Dec 2024; 130k). The dataset includes 2,500 features with pervasive missingness (49% on average) and highly heterogeneous predictive strength (e.g., widely varying information value across features), closely mirroring production settings. Unlabeled samples used for self-supervised pre-training are drawn from the same time window as the labeled training split. Additional dataset details are provided in Appendix A.

4.2 Experimental Settings

Model Variants. We define several MaskTab variants for ablation, scaling, and deployment. `MaskTab-Base` serves as the default model, while `MaskTab-{S/M/L/XL}` scales capacity from 25M

to 200M parameters. For production settings, we report `MaskTab-Distill`, a Base-sized student distilled from the best-performing `MaskTab-L`. See Appendix B for details.

Implementation Details. We train `MaskTab-Base` end-to-end with our hybrid supervised objective, following the optimization protocol of Hoffmann et al. (2022). We use a batch size of 2048 and a cosine learning-rate schedule with 100 warmup steps, the learning rate peaks at 1×10^{-4} and decays to 1×10^{-5}. To study scaling laws efficiently, we progressively scale along three axes and stop once performance saturates on each axis, substantially reducing training cost. When increasing the feature dimension, we add features in descending order of their Information Value (IV) scores. Models in our scaling study are trained with a two-stage schedule: (1) pre-training on large-scale labeled and unlabeled data with a learning rate scaled by model size, followed by (2) fine-tuning on labeled data only for stability. Finally, we conduct knowledge distillation by aligning the embedding representations of a high-performance `MaskTab-L` teacher model with those of a compact `MaskTab-Base` student model. All experiments were conducted using 8 NVIDIA A100 GPUs. Detailed hyperparameters, implementation procedures, and reproduction details for all baselines are provided in Appendix C.

4.3 Baseline evaluation

Baseline. To thoroughly evaluate the effectiveness of our proposed method on the TabRed dataset, we benchmark it against two prominent categories of models: Gradient Boosted Decision Trees (GBDTs) and Deep Learning models, as shown in Table 4.

For the private dataset CreditRisk, we trained XGBoost(Chen and Guestrin, 2016) classifiers on each feature subset $\mathcal{G}_m$ ($d_m = 500m$, $m = 1, \ldots, 5$). Models were fitted on $\mathcal{D}_{\text{train}}$, hyperparameters were selected via grid search on $\mathcal{D}_{\text{val}}$, and the best configurations were evaluated on the held-out $\mathcal{D}_{\text{test}}$, as detailed in Appendix A.

Figure 1 shows diminishing returns once the feature count exceeds 1,500 (i.e., for $m \geq 3$), suggesting a dimensionality bottleneck. While XGBoost still gains slightly from additional features, the large train–test KS gap (Fig. 4) indicates poor OOD generalization and potential performance decay after deployment.

Evaluation Protocol. We evaluate temporal robustness via out-of-time (OOT) testing by splitting the test set into monthly subsets. We report the mean Area Under the ROC Curve (AUC) and Kolmogorov-Smirnov (KS) across months to measure ranking quality and score stability under time shifts.

4.4 Overall Performance

The comparative results of MaskTab and other methods on TabRed are presented in Table 1. By calculating the average ranking across 8 datasets, our method achieves an average rank of **2.3**. This performance surpasses XGBoost, the preeminent algorithm among GBDT methods, and significantly exceeds that of other deep tabular models.

Table 2 reports results on the CreditRisk dataset, demonstrating the effectiveness of the MaskTab series. `MaskTab-Base`, trained solely on labeled data with a carefully selected set of interpretable features, significantly outperforms existing state-of-the-art methods. It achieves a 0.69% improvement in OOT AUC and a 0.81% improvement in KS over XGBoost. Our best-performing model from the extended experiments, `MaskTab-L`, shows an even greater improvement: a 5.04% gain in AUC and an 8.28% gain in KS over XGBoost, fully highlighting the advantage of large-scale representation learning. To balance performance and deployability, we apply knowledge distillation to transfer the representations learned by `MaskTab-L` into a compact student model, `MaskTab-Distill`. This variant shares the same architecture and input format as `MaskTab-Base`. It achieves a 2.55% improvement in AUC and a 4.85% improvement in KS over XGBoost, showing significant gains over the original base model while maintaining low inference latency and compatibility with standard interpretability tools such as SHAP. These results indicate that `MaskTab-Distill` is the recommended choice for deployment, achieving an optimal balance between accuracy, efficiency, and interpretability in real-world applications.

4.5 Ablation Study

To quantify the contribution of each design choice in MaskTab, we perform ablations on industrial CreditRisk dataset by progressively adding components to a vanilla Transformer encoder baseline (zero imputation + task-only training).

As shown in Table 3, introducing learnable mask embeddings together with hybrid supervised pre-training (task loss + masked reconstruction) yields the largest improvement over the baseline (+5.04%

Table 1: Comparing MaskTab with current Gradient Boosted Decision Tree (GBDT) methods and deep tabular models using the TabReD benchmark. The entries highlighted in bold indicate the best-performing models.

Method	Classification (ROC AUC ↑)			Regression (RMSE ↓)					Avg. Rank
	Homesite Insurance	Ecom Offers	HomeCredit Default	Sberbank Housing	Cooking Time	Delivery ETA	Maps Routing	Weather	
XGBoost(Chen and Guestrin, 2016)	0.9601	0.5763	0.8670	0.2419	0.4823	0.5468	**0.1616**	1.4671	4.4
LightGBM(Ke et al., 2017)	0.9603	0.5758	0.8664	0.2468	0.4826	0.5468	0.1618	1.4625	5.5
CatBoost(Prokhorenkova et al., 2018)	0.9606	0.5596	0.8621	0.2482	0.4823	**0.5465**	0.1619	1.4688	5.6
SNN(Klambauer et al., 2017)	0.9492	0.5996	0.8551	0.2858	0.4838	0.5544	0.1651	1.5649	10.5
DCNv2(Wang et al., 2020)	0.9392	0.5955	0.8466	0.2770	0.4842	0.5532	0.1672	1.5782	11.4
ResNet(Gorishniy et al., 2021)	0.9469	0.5998	0.8493	0.2743	0.4825	0.5527	0.1625	1.5021	8.0
MLP-PLR(Gorishniy et al., 2022)	0.9621	0.5957	0.8568	0.2438	0.4812	0.5527	**0.1616**	1.5177	4.4
Trompt(Chen et al., 2023)	0.9546	0.5792	0.8381	0.2596	0.4834	0.5563	0.1652	1.5722	11.0
FT-Transformer(Gorishniy et al., 2021)	0.9622	0.5775	0.8571	0.2440	0.4820	0.5542	0.1625	1.5104	6.4
TabR(Gorishniy et al., 2024)	0.9522	0.5850	0.8484	0.2851	0.4825	0.5541	0.1637	**1.4622**	8.6
TabNet (Arik and Pfister, 2021)	0.9531	0.5855	0.7701	0.2828	0.4813	0.5567	0.1651	1.5877	10.5
TransTab(Wang and Sun, 2022)	0.9564	0.5868	0.8498	-	-	-	-	-	-
CM2(Ye et al., 2024)	0.9560	0.5890	0.8392	**0.2287**	0.4838	0.5569	0.1638	1.5339	9.0
MaskTab(Base)	**0.9635**	**0.6069**	**0.8698**	0.2345	**0.4806**	0.5486	0.1618	1.4861	**2.3**

Table 2: Performance comparison across models on the CreditRisk dataset.

Model	Type	#Params (Est.)	#Features	ROC AUC ↑	KS ↑
XGBoost(Chen and Guestrin, 2016)	GBDT	-	500	70.80	33.17
LightGBM(Ke et al., 2017)	GBDT	-	500	69.49	32.04
TabNet(Arik and Pfister, 2021)	Seq-to-Seq	29.71M	500	65.81	26.87
MLP-PLR(Gorishniy et al., 2022)	Feedforward	37.24M	500	66.50	28.60
FT-Transformer(Gorishniy et al., 2021)	Transformer	31.52M	500	68.32	28.32
CM2(Ye et al., 2024)	Transformer	26.01M	500	68.35	29.22
MaskTab-Base	Transformer	25.16M	500	71.49	33.98
MaskTab-L	Transformer	134.22M	2,000	**75.84**	**41.45**
MaskTab-Distill	Transformer	25.16M	500	73.35	38.02

Table 3: The effectiveness of each component in MaskTab-Base.

Module Additions	ROC AUC ↑	KS ↑
Vanilla baseline	64.72	23.71
+ Mask Embedding	69.76 (**+5.04%**)	31.07 (**+7.36%**)
+ Twin Networks	70.90 (**+1.14%**)	32.41 (**+1.34%**)
+ MoE	71.49 (**+0.59%**)	33.98 (**+1.57%**)

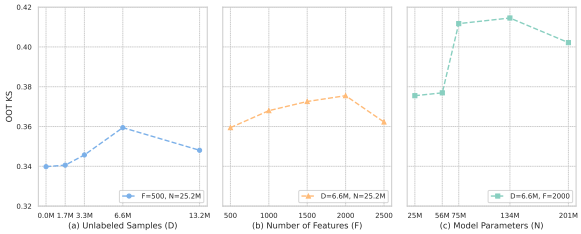

Figure 3: Scaling Law Analysis. (a) Scaling up the amount of unlabeled data, with the number of features fixed at 500 and the model parameters fixed at 25.2M. (b) Scaling up the number of features, with the amount of unlabeled data fixed at 6.6M and the model parameters fixed at 25.2M. (c) Scaling up the model parameters, with the amount of unlabeled data fixed at 6.6M and the number of features fixed at 2000.

AUC, +7.36% KS), highlighting the importance of explicitly modeling missingness and aligning pre-training with the downstream objective. Replacing joint training with the proposed twin-path training further improves performance (+1.14% AUC, +1.34% KS), confirming that separating reconstruction (masked input) from prediction (natural missingness) mitigates masking-induced shift. Finally, adding the MoE reconstruction head provides additional gains (+0.59% AUC, +1.57% KS), suggesting that expert routing improves reconstruction capacity via dynamic feature grouping.

4.6 Scaling Law Analysis

Impact of the unlabeled data size. We first study data scaling by fixing the input feature budget to 500 and the model capacity to 25.2M parameters, and varying only the amount of unlabeled data. Concretely, we pre-train on the 330K labeled training split mixed with unlabeled samples at 0×, 5×, 10×, 20×, and 40× the labeled set, and then

fine-tune on labeled data only to ensure a fair downstream comparison. As shown in Figure 3, performance improves monotonically up to 20× unlabeled data, where OOT KS reaches 0.3594 (+1.9% over the labeled-only baseline). Scaling further to 40× lowers KS to 0.3480, indicating diminishing returns and potential saturation under the current feature and capacity settings. In subsequent experiments, we therefore also examine scaling along the feature dimension and model size, which are complementary levers for better exploiting large unlabeled corpora.

Impact of the number of features. Following the data scaling study, we fix the unlabeled-to-labeled ratio at 20× and scale the feature dimension from 500 to 2,500 (step size 500), keeping all other settings unchanged. As shown in Figure 3, OOT KS increases with more features up to 2,000, peaking at 0.3754 (+1.6% over 500 features), and then drops at 2,500 (0.3653), indicating saturation. Overall, MaskTab effectively leverages additional high-dimensional, sparse features, demonstrating robustness on industrial tabular data with missingness and long-tailed feature distributions.

Impact of model parameters. The above results suggest diminishing returns when scaling either unlabeled data or feature dimension under a fixed architecture. We therefore scale model capacity under the best configuration (unlabeled-to-labeled ratio = 20×, 2,000 features), increasing parameters from `MaskTab-Base` (25.16M) to `MaskTab-XL` (201.32M). Figure 3 shows that KS improves monotonically up to `MaskTab-L` (134M), reaching 0.4145 (+3.91% over the previous best 0.3754), but drops at `MaskTab-XL` (0.4021), again indicating saturation. Overall, MaskTab exhibits consistent scaling behavior across data, features, and model size, providing a practical recipe for improving tabular models by jointly scaling these three factors.

4.7 Knowledge Distillation for Deployment

As shown in Figure 1, `MaskTab-L-2000` (134.21M) achieves the best KS among the MaskTab variants, but is less suitable for production due to its high inference cost and limited interpretability with 2000 sparse features. We therefore distill `MaskTab-L-2000` into an efficient and auditable student, `MaskTab-Base-500`, by aligning their representations during training, yielding `MaskTab-Distill-500`. Distillation improves KS by 4.04% over `MaskTab-Base-500`, and achieves a 9.3× inference speedup while using only 500 interpretable features.

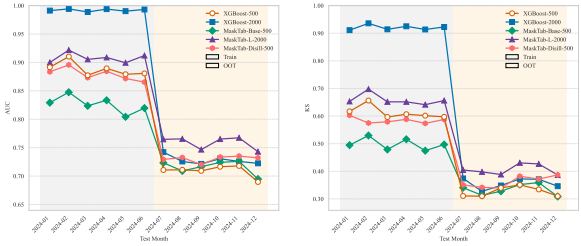

Figure 4: OOD Analysis. Monthly AUC (left) and KS (right) over time. Models are trained on 2024-01–2024-06 and evaluated on the OOT period 2024-07–2024-12; a smaller train–OOT gap indicates stronger robustness to distribution shift and better production generalization.

4.8 OOD Analysis

Figure 4 compares training vs. out-of-time (OOT) KS to measure robustness under temporal distribution shift. GBDT baselines (e.g., XGBoost) exhibit a large generalization gap (training KS > 0.60 vs. OOT KS ≈ 0.35), indicating substantial overfitting to period-specific signals. In contrast, `MaskTab-Base` markedly reduces this gap, and `MaskTab-Distill` further stabilizes it, yielding the smallest train–OOT discrepancy. These results suggest that our pre-training and distillation pipeline improves robustness to OOD drift, leading to more reliable performance after deployment.

5 Conclusion

In this paper, we present MaskTab, a unified pre-training framework for industrial tabular data that are high-dimensional, riddled with missing entries, and rarely labeled at scale. MaskTab uses learnable tokens to encode both masked and naturally missing values, and adopts twin-path hybrid pre-training that couples task supervision with masked reconstruction. An MoE reconstruction head further allocates capacity across heterogeneous features. Across large-scale industrial benchmarks, MaskTab achieves state-of-the-art performance.

Importantly, we establish clear scaling trends: performance improves predictably as we scale unlabeled data, feature dimension, and model capacity, offering practical guidance for compute allocation. Finally, our distillation pipeline transfers these gains to compact, interpretable students for low-latency deployment.

Limitations

Our scaling study is not exhaustive. Due to resource constraints, we adopt a greedy strategy that scales one dimension at a time (unlabeled data, feature count, or model size), which reveals consistent gains but does not yield a formal joint scaling law, a larger factorial exploration is needed. In addition, extending MaskTab to richer modalities (e.g., more substantial text signals) and temporal/tabular time-series data remains future work.

References

Sercan Ö Arik and Tomas Pfister. 2021. TabNet: Attentive interpretable tabular learning. In *AAAI*.

Kuan-Yu Chen, Ping-Han Chiang, Hsin-Rung Chou, Ting-Wei Chen, and Tien-Hao Chang. 2023. Trompt: Towards a better deep neural network for tabular data. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 4392–4434. PMLR.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *SIGKDD*.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, and 1 others. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *NAACL*.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. In *NeurIPS*.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. 2024. Tabr: Tabular deep learning meets nearest neighbors in 2023. In *ICLR*.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. In *NeurIPS*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.

Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. 2019. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 384–394.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *NeurIPS*.

Weijiang Lai, Beihong Jin, Jiongyan Zhang, Yiyuan Zheng, Jian Dong, Jia Cheng, Jun Lei, and Xingxing Wang. 2025. Exploring scaling laws of ctr model for online performance improvement. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 114–123.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L Caterini. 2024. Tabdpt: Scaling tabular foundation models. *arXiv preprint arXiv:2410.18164*.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. Catboost: unbiased boosting with categorical features. In *NeurIPS*.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, and 1 others. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Ivan Rubachev, Nikolay Kartashev, Yury Gorishniy, and Artem Babenko. 2024. Tabred: A benchmark of tabular machine learning in-the-wild. *arXiv*, 2406.19380.

Ruoxi Wang, Rakesh Shivanna, Derek Z. Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. 2020. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. *arXiv*, 2008.13535v2.

Zifeng Wang and Jimeng Sun. 2022. TransTab: Learning transferable tabular Transformers across tables. In *NeurIPS*.

Qingren Yao, Chao-Han Huck Yang, Renhe Jiang, Yuxuan Liang, Ming Jin, and Shirui Pan. 2024. Towards neural scaling laws for time series foundation models. *arXiv preprint arXiv:2410.12360*.

Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. 2024. Towards cross-table masked pretraining for web data mining. In *WWW*.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. 2022. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113.

A Dataset Details

A.1 TabRed

We employed the TabReD benchmark, which exclusively consists of tabular data obtained from real-world industrial applications. This benchmark comprises 8 datasets, including 3 classification tasks and 5 regression tasks. As outlined in Table 4, each dataset is characterized by a substantial number of samples and features. Furthermore, we evaluated the rates of missing features for each dataset and found that most exhibit varying degrees of incompleteness. Additionally, these datasets were divided into training, validation, and testing sets based on temporal criteria. These characteristics are frequently encountered in industrial applications, rendering this benchmark particularly suitable for assessing model performance in such contexts.

A.2 CreditRisk

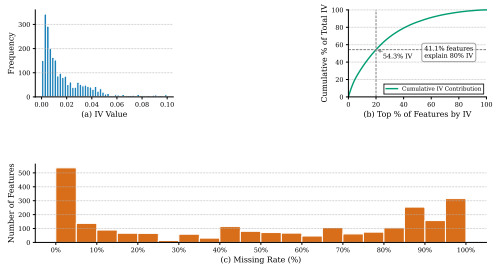

Figure 5: Data Analysis of CreditRisk. Subfigure (a) displays the distribution of IV values across all features, showing a long-tail trend. Subfigure (b) illustrates the cumulative IV values as the feature proportion increases. Subfigure (c) presents the distribution of feature counts with varying missing rates.

Temporal Partitioning. We proposed a proprietary credit risk prediction dataset named CreditRisk, as shown in Table 4. It is characterized by 6.4×10^5 labeled and 1.3×10^7 unlabeled instances from 2024, which are partitioned temporally to avoid leakage. The data contains a large set of 2,500 features and exhibits pervasive missingness with an average rate of 49%, thereby reflecting real-world production constraints. We utilize records spanning the entire year of 2024. Labeled data is partitioned monthly into: Training (Jan–Jun 2024): $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{\text{train}}}$, $|\mathcal{D}_{\text{train}}| \approx 3.3 \times 10^5$. Validation (Jul–Sep 2024): $\mathcal{D}_{\text{val}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{\text{val}}}$, $|\mathcal{D}_{\text{val}}| \approx 1.8 \times 10^5$. Test (Oct–Dec 2024): $\mathcal{D}_{\text{test}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{\text{test}}}$, $|\mathcal{D}_{\text{test}}| \approx 1.3 \times 10^5$. Unlabeled data used for self-supervised pre-training are drawn from the same six-month window as the training set, $\mathcal{D}_{\text{unsup}} = \{\mathbf{x}_j\}_{j=1}^{M}$ with $M \approx 1.3 \times 10^7$, thereby eliminating any temporal leakage.

Scale and dimensionality. The total feature space contains $d = 2{,}500$ attributes ($\mathcal{F} = f_k{}_{k=1}^{d}$). With labeled instances numbering in the hundreds of thousands and unlabeled instances in the tens of millions, this dataset thus provides a platform for scaling up research.

Pervasive missingness. For each feature (k) we define a missing-rate on the training set

$$\eta_k = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i=1}^{|\mathcal{D}_{\text{train}}|} \mathbb{I}!\big(v_k^{(i)} = \perp\big), \eta_k \in [0.0, 1.0),$$

where $\perp$ denotes a missing value. The average missing rate across all features is $\bar{\eta} \approx 0.49$, the distribution of η_k is shown in Figure 5 (distribution of feature missing rates).

Feature-selection protocol. We adopt the conventional pipeline of ranking raw attributes by a univariate importance score $\mathcal{I}(f_k)$ (e.g., information value, IV). After sorting the $d = 2{,}500$ features in descending order of $\mathcal{I}$, we partition them into five incremental groups:

$$\mathcal{G}_m = \{f_k \mid \text{rank}(f_k) \le m \times 500\}, \quad m \in \{1, 2, 3, 4, 5\},$$

so that the m-th group $\mathcal{G}_m$ contains the top $d_m = 500m$ features. This stratification serves two purposes. First, it provides a series of baseline models built on increasingly larger feature subsets, allowing us to quantify the marginal benefit of adding more variables. Second, by progressively introducing groups of lower-importance features we can assess whether self-supervised pre-training amplifies the model's ability to exploit weak signals. The empirical distribution of IV values across all features is visualised in Figure 5.

Table 4: The table presents details of nine datasets, eight from TabRed and one proprietary dataset named CreditRisk. For each dataset, we report the number of samples (# samples), number of numerical features (#num features), and number of categorical features (#cat features). We also analyze the extent of missingness in each dataset by categorizing feature missing rates into three bins and reporting the proportion of samples falling into each bin.

task type	dataset name	source	#samples	#num features	#cat features	sample proportion at feature missing rates		
						(0%, 33%]	(33%, 66%]	(66%, 100%]
Classification	Ecom Offers (EO)	TabRed	160,057	113	6	0.00%	0.00%	0.00%
	Homesite Insurance (HI)	TabRed	260,753	253	46	89.93%	0.00%	0.00%
	HomeCredit Default (HCD)	TabRed	381,664	612	84	57.16%	36.93%	5.91%
	CreditRisk (CR)	Private	13 million	2475	25	6.75%	89.17%	4.08%
Regression	Sberbank Housing (SH)	TabRed	28,321	365	27	100.00%	0.00%	0.00%
	Cooking Time (CT)	TabRed	319,986	186	6	99.10%	0.00%	0.00%
	Delivery ETA (DE)	TabRed	350,516	221	2	91.50%	3.71%	0.00%
	Maps Routing (MR)	TabRed	279,945	984	2	97.08%	2.82%	0.10%
	Weather (W)	TabRed	423,795	100	3	0.00%	2.21%	0.03%

Table 5: Architecture and training configuration of the MaskTab model series. All models share a Transformer backbone with progressively scaled capacity and input data.

Model	#Params(Non-Emb)	Layers	Number Heads	Key/Value Size	d_{model}
MaskTab-Base	$25.16M$	6	8	64	512
MaskTab-S	$56.62M$	6	12	64	768
MaskTab-M	$75.50M$	8	12	64	768
MaskTab-L	$134.22M$	8	16	64	1024
MaskTab-XL	$201.33M$	12	16	64	1024
MaskTab-Distill	$25.16M$	6	8	64	512

B Model Variants Details

MaskTab-Base. Trained on the `CreditRisk` dataset of interpretable features using labeled data and a mixed objective (e.g., reconstruction and classification). Serves as the baseline and student model.

MaskTab-{S/M/L/XL}. Scaled versions extending along model parameters, the number of features, and the volume of unlabeled data. `MaskTab-L` achieved the best performance among the scaled-up variants under the specific data and feature set used in this work. Pretrained on tens of millions of unlabeled samples, it was consequently employed as the teacher model to guide other variants via knowledge distillation. A summary of the hyperparameter settings for the MaskTab model family is presented in Table 5.

MaskTab-Distill. A distilled version of `MaskTab-Base`. It retains the identical model architecture and parameter count as the `MaskTab-Base` but is trained to mimic the outputs and representations of the larger `MaskTab-L`. This process yields higher accuracy while preserving the base model's low latency and interpretability, making it ideal for production deployment.

C Implementation Details

C.1 Model Comparison

To situate our approach within the broader landscape of tabular data modeling, we compare MaskTab with a range of representative baselines, including Gradient Boosted Decision Trees (GBDTs), deep learning models, and recent pre-training approaches, as detailed in Table 1 and Table 2. All models are trained and evaluated on identical data splits.

Table 6: The hyper-parameter optimization space for MaskTab-Base.

Hyper-parameter	Distribution
Learning rate	$\{1 \times 10^{-5}, 3 \times 10^{-5}, 1 \times 10^{-4}\}$
Number of Transformer blocks	{3, 4, 5}
Number of attention heads	{8, 16}
Feature embedding dimension	{64, 128, 256}
Batch size	{128, 256, 512, 2048}
Weight Decay	$\{0, 1 \times 10^{-2}, 1 \times 10^{-4}\}$
Dropout rate	{0.0, 0.15, 0.3}

The parameter search spaces for MaskTab-Base, XGboost, and LightGBM are listed in Table 6, Table 7, and Table 8, respectively.

In addition to the TabReD benchmark, we have implemented some additional DNN-based methods, with the experimental settings for each method as

Table 7: The hyper-parameter optimization space for XGboost.

Hyper-parameter	Distribution
Learning rate	{0.05, 0.1, 0.15, 0.2}
N_estimators	Const(1000)
Max depth	{4, 5, 6, 7}
Reg_lambda	{10, 50, 100, 500, 1000}
Subsample	{0.3, 0.5, 0.7, 0.9, 1.0}

Table 8: The hyper-parameter optimization space for LightGBM.

Hyper-parameter	Distribution
Boosting	{gbdt, goss, rf, dart}
Learning rate	{0.05, 0.1, 0.15, 0.2}
Num_leaves	Const(31)
Min_data_in_leaf	Const(20)
N_estimators	Const(500)
Max depth	{4, 5, 6, 7}
Feature_fraction	Const(1)
Max_bin	Const(255)
Min_data_in_bin	Const(3)
Bin_construct_sample_cnt	Const(200000)

follows:

TabNet: Use the official implementation[1]. The width of the decision prediction layer and the attention embedding for each mask is 8. The number of steps is 3, the learning rate is 0.02, and the early stopping patience is set to 20 out of a maximum of 200 epochs.

TransTab: Use the official implementation[2]. The model consists of 2 transformer layers. The dimensions of the hidden embedding and the feed-forward layers are 128 and 256, respectively, and the number of attention heads is 8. For training the downstream task, we use a batch size of 64, a learning rate of 1e-4, a dropout rate of 0, and an early stopping patience of 20 out of a maximum of 200 epochs.

CM2: Use official implementation[3]. The token embedding dimension is 128, the hidden dimension of the feed-forward layers is 256, and the self-attention module has 8 heads. 3 transformer layers are used. It is trained with a batch size of 64, a learning rate of 1e-4, and a dropout rate of 0.15. The maximum training epoch is 200, and the patience value for early stopping is set to 10. We fine-tuned using pre-trained weights (*CM2-v1*) and also fine-tuned from scratch, subsequently selecting the optimal results. For our private industrial dataset CreditRisk, fine-tuning using the pre-trained weights (CM2-v1) achieved an AUC of 0.8920 and a KS of 0.6080. In contrast, fine-tuning from scratch resulted in an AUC of 0.8937 and a KS of 0.6111.

MaskTab-Base: Trained end-to-end using a hybrid supervised strategy that combines pre-training and classification tasks. We adopt the optimization setup from Hoffmann et al. (2022), using a batch size of 2048 and a cosine annealing schedule with a 100-step warmup. The learning rate peaks at 1×10^{-4} and decays by a factor of 10. The model is evaluated after a single training stage without additional fine-tuning. All experiments were conducted using 8 NVIDIA A100 GPUs.

To quantify the contribution of each component in `MaskTab-Base`, we perform ablation studies by incrementally incorporating key modules: Parameter-Shared Mask Embedding, Siamese Network, and Scalable MoE Reconstruction Head.

C.2 Scaling Laws.

We empirically investigate scaling laws in tabular data learning by varying three factors: the amount of unlabeled data, the number of features, and the model parameters.

For data scaling, as introduced in Section 4.1, we held constant a labeled training dataset of 3.3×10^5 samples while progressively increasing the volume of unlabeled data. This resulted in labeled-to-unlabeled data ratios of 1:5, 1:10, 1:20, and 1:40.

For feature scaling, we adhered to the experimental protocol outlined in Section 4.1. Features are ranked by Information Value (IV) scores. We then expand the feature set in increments of 500 features up to 2500 features. This process mimics real-world scenarios where additional features typically have diminishing predictive power.

For model scaling, we introduce `MaskTab-{S/M/L/XL}`, a model family with increasing parameters. We adopt a two-stage training procedure to ensure stable learning across scales:

- **Pretraining:** Mixed-supervision on both labeled and unlabeled data. The learning rate is scaled as:

$$\mathrm{LR}_{\mathrm{scaled}} = \mathrm{LR}_{\mathrm{base}} \times \frac{1}{\sqrt{N/N_{\mathrm{base}}}},$$

  where N is the parameter count of the current model and N_{base} is that of `MaskTab-Base`.

- **Fine-tuning:** Supervised training on labeled data only, with learning rate reduced to 1×10^{-5}.

[1] https://github.com/dreamquark-ai/tabnet
[2] https://github.com/RyanWangZf/transtab
[3] https://github.com/Chao-Ye/CM2

C.3 Knowledge Distillation.

To meet the real-time and interpretability constraints of risk control systems, we distill knowledge from a high-performance teacher model to a compact student. The teacher is a `MaskTab-L` model trained on 2,000 features, which was the best performer in our scaling-up experiments. The student is a `MaskTab-Base` model restricted to 500 interpretable features. The distillation process first caches the embedding vectors generated by the teacher model on the training set. It then trains the student model using these cached embeddings as soft targets, with a distillation loss enforcing representation alignment (Section 3.6).