TABGIFT: A CONTEXT-EFFICIENT FOUNDATION MODEL FOR TABULAR DATA GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular data generation aims to synthesize realistic samples from limited observations while faithfully capturing the underlying data distribution. Tabular foundation models (TFMs), pretrained on diverse tabular tasks and adapted through in-context learning, have demonstrated strong predictive capabilities, suggesting the potential of extending this paradigm to tabular data generation. However, the strong predictive performance of TFMs often relies on sufficiently large context sets, whereas tabular generation is typically needed precisely when target data are scarce. Extending the foundation-model paradigm to generation therefore requires reliable distribution inference from only a limited context. To this end, we propose TABGIFT, a *context-efficient* tabular foundation model enabling both efficient distribution modeling and effective generation from only a few context samples. TABGIFT maps heterogeneous tables into a shared latent space with a pretrained tabular encoder, where an in-context diffusion model learns to denoise latent samples conditioned on a small set of clean examples. This design transfers generative priors across datasets and enables distribution inference from limited context, while a lightweight dataset-specific MLP decoder maps generated representations back to the original data space. Experiments on 7 classification and 11 regression datasets show that TABGIFT consistently outperforms representative baselines in synthetic data quality, with particularly strong gains under limited-context settings, while achieving substantially faster adaptation and generation.

1 INTRODUCTION

Tabular data are ubiquitous in real-world applications such as finance (Zheng et al., 2020; Clements et al., 2020; Huang et al., 2023), healthcare (Gong et al., 2020; Yang et al., 2021), and scientific discovery (Du et al., 2021; Wu et al., 2021; Zhang et al., 2023). However, high-quality tabular data are often limited by privacy constraints, restricted accessibility, and costly collection or annotation. Tabular data generation (Mehrabi et al., 2021; Dastin, 2022) therefore provides an important means to alleviate data scarcity by synthesizing realistic samples that preserve the underlying data distribution while reducing reliance on direct access to sensitive or limited real-world data.

Existing tabular generation methods are mainly non-parametric (Chawla et al., 2002; He et al., 2008; Zhang et al., 2018) or parametric (Xu et al., 2019; Shi et al., 2024; Zhang et al., 2024). The former depend on observed samples for resampling or interpolation, while the latter require dataset-specific training and tuning. These limitations motivate tabular foundation models, which aim to amortize learning across datasets and adapt to new tables with minimal target-specific optimization.

Tabular foundation models (Hollmann et al., 2025; Jäger et al., 2026; Grinsztajn et al., 2026; 2025) provide a promising solution to these limitations. By pretraining on diverse tabular tasks and adapting through in-context learning, they can achieve strong predictive performance on unseen datasets without dataset-specific retraining or hyperparameter tuning. Their ability to rapidly transfer across datasets naturally suggests a promising direction for tabular generation, leading to a direct question: *can tabular foundation models also be extended to tabular data generation?*

However, extending TFMs from prediction to generation introduces a distinct challenge in context efficiency. Unlike prediction, which only requires task-specific outputs conditioned on observed features, generation must infer the joint distribution of all variables from the provided context, making reliable generation more demanding under limited observations. Yet relying on large context sets is

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.

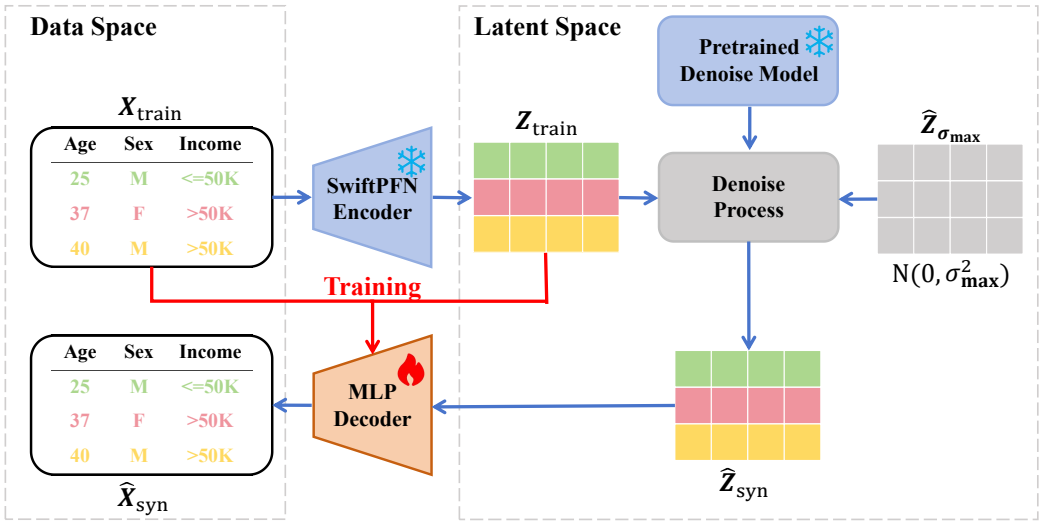


Figure 1: TABGIFT consists of a data space and a latent space: (1) Encoding: Given the real training table X_{train}, the SwiftPFN Encoder maps heterogeneous samples into a unified 192-dimensional latent space, producing the latent embeddings Z_{train}. (2) Latent-space generation: The real-data latent latent embedding Z_{train} is used as context. Starting from Gaussian noise, a pretrained frozen denoising model generates new latent embeddings $\hat{Z}_{\text{syn}}$ through an iterative denoising process. (3) Decoding: A lightweight MLP Decoder is trained per target dataset to reconstruct the generated latent embeddings $\hat{Z}_{\text{syn}}$ back into the data space, yielding synthetic data $\hat{X}_{\text{syn}}$. Throughout the pipeline, only the MLP Decoder requires dataset-specific training, while the SwiftPFN Encoder and the denoising model are pretrained and kept frozen during inference.

fundamentally at odds with the data-scarce settings (Wang et al., 2026b; Park & Ko, 2026; Lin et al., 2026; Shen et al., 2026) where synthetic data are most valuable. Meanwhile, accurately capturing dependencies within the context often requires fine-grained interactions among features and samples, whose computational cost grows rapidly with context length and table dimensionality (dpg, 2025). Therefore, an effective generative TFM must not only infer target distributions from a small context, but also model that context efficiently.

To address these challenges, we propose TABGIFT, a *context-efficient foundation model* for tabular data generation. TABGIFT builds a shared latent space to unify heterogeneous tables, performs in-context generation in this space to infer target distributions from limited observations, and uses lightweight dataset-specific decoding for efficient adaptation. Together, these designs enable high-quality generation from short contexts with low computational overhead.

The generation pipeline is illustrated in Fig 1. Specifically, TABGIFT first employs a pretrained tabular encoder to map rows from heterogeneous tables into a shared latent space, enabling a unified generative model across datasets with different schemas. In this space, we develop an in-context diffusion model that conditions noisy latent samples on a small set of clean context examples and performs denoising through efficient row-level interactions. This formulation allows the diffusion model to reuse generative knowledge across datasets while adapting the generated distribution to the provided context. Finally, a lightweight dataset-specific MLP decoder maps the generated latent representations back to the data space, requiring only minimal target-specific adaptation. Experiments on 7 classification and 11 regression datasets show that TABGIFT consistently outperforms representative baselines in synthetic data quality, with particularly strong gains under limited-context settings and substantially improved generation efficiency. Our contributions are threefold:

- We highlight the importance of context efficiency in foundation models for tabular generation, showing that long-context dependence fundamentally conflicts with the data-scarce settings where synthetic data generation is most valuable.

- We propose TABGIFT, a context-efficient tabular foundation model that combines transferable distributional priors, efficient row-level modeling, and lightweight dataset-specific decoding for fast generation from limited context.
- We conduct extensive experiments across 18 classification and regression datasets, demonstrating that TABGIFT consistently improves synthetic data quality under limited-context settings while achieving substantially higher generation efficiency.

2 RELATED WORK

2.1 TABULAR GENERATION MODELS

Early work on tabular data generation primarily focused on improving downstream predictors rather than learning a high-fidelity data generator. Representative methods such as SMOTE (Chawla et al., 2002) and ADASYN (He et al., 2008) address class imbalance by synthesizing minority samples, but they do not aim to match the full joint distribution of the original table.

A subsequent line of work instead fits expressive generative models to observed samples. These methods can be broadly grouped into two categories. The first adapts image-inspired generators to tables, including GAN-based models such as ADSGAN (Yoon et al., 2020) and CTGAN (Xu et al., 2019), VAE-based approaches such as TVAE (Xu et al., 2019), and diffusion-based generators such as TabDDPM (Kotelnikov et al., 2023), TabDiff (Shi et al., 2024), and TabSyn (Zhang et al., 2024). The second develops table-specific families: Bayesian networks (Ankan & Textor, 2024) factorize the joint distribution along a directed acyclic graph, normalizing flows (Durkan et al., 2019) provide tractable likelihoods and efficient sampling, and tree-based alternatives such as Adversarial Random Forests (Watson et al., 2023) offer non-parametric density estimation and capture feature interactions. While these approaches can model complex distributions, they are typically trained per dataset and often degrade when training data are scarce.

2.2 TABULAR FOUNDATION MODELS

Tabular foundation models have evolved rapidly from the Prior-Fitted Network (PFN) paradigm introduced by TabPFN (Hollmann et al., 2023; Grinsztajn et al., 2025; Hollmann et al., 2025; Jäger et al., 2026; Grinsztajn et al., 2026), which pretrains on synthetic tasks and performs prediction on unseen datasets through in-context learning. Building on this framework, TabICL (Qu et al., 2025) further improves scalability and representation learning by introducing column-aware modeling and a multi-stage pretraining strategy, making foundation models applicable to larger and more heterogeneous tables. More recently, the field has expanded considerably, with models such as TabFM (Kong & Das, 2026), EXAONE (Eo et al., 2026), Mitra-2 (Tao et al., 2026), LimiX (lim, 2026), Xiaomi-TabLDM (Wang et al., 2026a), and newer versions of TabPFN exploring different directions in model scale, task coverage, architectural design, and pretraining strategies. Together, these developments have substantially advanced the capability and generality of tabular foundation models.

Compared with predictive modeling, tabular foundation models for data generation remain relatively underexplored. TabPFGen (Ma et al., 2023) and TabEBM (Margeloiu et al., 2024) combine TabPFN with energy-based modeling (Grathwohl et al., 2020) to achieve foundation model generation ability. TabFORGE (Jiang et al., 2026) leverages pretrained causality-aware tabular representations for generative modeling, while DiffICL (Han et al., 2026) formulates tabular synthesis as an in-context learning problem to infer target distributions from limited examples. Despite these advances, existing methods still rely on relatively long contexts or costly interaction modeling, leaving context-efficient and computationally efficient generation insufficiently explored.

3 METHOD

3.1 OVERVIEW

In this section, we introduce the architecture of TABGIFT and how to use TABGIFT to generate new samples. TABGIFT can be viewed as a latent diffusion model (LDM) (Lin et al., 2025; Zhu et al.,

2025; cor, 2025) consisting of three components: (1) an encoder obtained from the general tabular prediction model SwiftPFN (Liu & Ye, 2026), which leverages its pretrained generalization ability on tables to map heterogeneous tabular data into a unified latent space; (2) a score-based diffusion model (Zhang et al., 2024) trained in the latent space via in-context learning, where latent embeddings of real samples serve as context to help the denoising process for noisy embeddings, enabling sampling from random noise; and (3) a dataset-specific MLP decoder that maps the generated latent embeddings from the latent space back to the concrete data space.

3.2 SETUP

We consider a tabular dataset $X=\{\boldsymbol{x}_i\}_{i=1}^{N}$, where N is the number of samples, $\boldsymbol{x}_i \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_F$ is the feature vector with F features. Features may be numerical or categorical.

Given a training set X_{train}, tabular data generation aims to learn a model p_θ that produces a synthetic dataset $\hat{X}_{\text{syn}}$ approximating the underlying distribution of X_{train}, while preserving marginal distributions and inter-feature dependencies and avoiding copying of training samples.

3.3 SWIFTPFN ENCODER

We use SwiftPFN as a general-purpose encoder to handle heterogeneous datasets, which is a foundation tabular prediction model. Using only sample-attention, SwiftPFN achieves performance comparable to Limix (lim, 2026) and TabPFNv2 (Hollmann et al., 2025) (which use sample-attention and feature-attention). In specific, SwiftPFN first maps both the training samples ($\boldsymbol{X}_{\text{train}}$ and $\boldsymbol{y}_{\text{train}}$) and the test samples ($\boldsymbol{X}_{\text{test}}$ only) to be predicted into the same dimension ($d = 192$). It then uses a 12-layer Transformer architecture with a causal mask (training samples can attend to each other, while test samples can only attend to training samples) to enable interactions between the training and test samples. This allows the embedding of a test sample to learn label information, and finally a prediction head outputs the prediction result. Formally, the prediction process of SwiftPFN can be written as the following equation:

$$\begin{aligned}\boldsymbol{y}_{\text{pred}} &= \text{SwiftPFN}(\boldsymbol{X}_{\text{test}}; \boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}}) \\ &= \text{PredHead}\Big(\text{Transformer}\big([\text{Embed}(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}}], \text{Embed}(\boldsymbol{X}_{\text{test}}))\big)[n:]\Big),\end{aligned}$$

where n denotes the length of the training samples, and $[n:]$ denotes taking the embedding corresponding to the test samples. Here, we can take the sample representation after the multiple Transformer layers and before the prediction layer as the latent embedding we desire. Since in generation tasks there is only $\boldsymbol{X}_{\text{train}}$, and we do not distinguish which column is the prediction target, to generate sample embeddings through SwiftPFN, we additionally construct an all-zero label $\boldsymbol{y}_{\text{train}} = \mathbf{0}_n$ and an all-zero test sample $\boldsymbol{X}_{\text{test}} = \mathbf{0}_{1\times F}$ to ensure the normal operation of the model.

$$\begin{aligned}\boldsymbol{Z}_{\text{train}} &= \text{SwiftPFNEncoder}(\mathbf{0}_{1\times F}; \boldsymbol{X}_{\text{train}}, \mathbf{0}_n) \\ &= \text{Transformer}\big([\text{Embed}(\boldsymbol{X}_{\text{train}}, \mathbf{0}_n), \text{Embed}(\mathbf{0}_{1\times F})]\big)[:n].\end{aligned}$$

3.4 IN-CONTEXT DIFFUSION MODEL

Diffusion models aim to train a denoiser that recovers the original data from noisy data and the noise level, their theory is already well established. TABGIFT trains a context-conditioned denoising network: by taking the latent embeddings of real samples as context, it learns to estimate the denoising direction for noisy latent embeddings, thereby avoiding dataset-specific training.

3.4.1 FORWARD TRAINING PROCESS

We adopt a score-based diffusion process as the main generative framework of TABGIFT, which has been widely validated in tabular generation tasks. Specifically, given the latent embedding $\boldsymbol{Z}_{\text{train}}$ obtained by the SwiftPFNEncoder, we divide it into two parts: $\boldsymbol{Z}_{\text{clean}}$ and $\boldsymbol{Z}_{\text{noisy}}$. $\boldsymbol{Z}_{\text{clean}}$ remains clean and serves as the in-context condition, while $\boldsymbol{Z}_{\text{noisy}}$ is treated as the target to be perturbed with noise of different magnitudes. The noising process is as follows:

$$\boldsymbol{Z}_\sigma = \boldsymbol{Z}_{\text{noisy}} + \sigma\epsilon, \quad \epsilon \sim \mathcal{N}(0,1),$$

where σ denotes the noise magnitude. It is typically sampled from a log-normal distribution $\ln \sigma \sim \mathcal{N}(p_{\text{mean}}, p_{\text{std}}^2)$. The denoiser takes $\boldsymbol{Z}_\sigma$ and the clean context $\boldsymbol{Z}_{\text{clean}}$ as input, and is trained to predict $\boldsymbol{Z}_{\text{noisy}}$. The training objective is

$$\ell_{\text{diff}} = \mathbb{E}_{\boldsymbol{Z}_{\text{noisy}},\sigma,\epsilon}\left[\frac{\sigma^2 + \sigma_{\text{data}}^2}{\sigma^2 \sigma_{\text{data}}^2}\left(\text{Denoiser}(\boldsymbol{Z}_\sigma, \sigma; \boldsymbol{Z}_{\text{clean}}) - \boldsymbol{Z}_{\text{noisy}}\right)^2\right].$$

The EDM-parametrized (Karras et al., 2022) denoiser estimates the clean latent:

$$\text{Denoiser}(\boldsymbol{Z}_\sigma, \sigma; \boldsymbol{Z}_{\text{clean}}) = c_{\text{skip}}(\sigma)\boldsymbol{Z}_\sigma + c_{\text{out}}(\sigma) D_\theta\left(c_{\text{in}}(\sigma)\boldsymbol{Z}_\sigma, c_{\text{noise}}(\sigma); \boldsymbol{Z}_{\text{clean}}\right).$$

Network D_θ is the in-context learning model that truly requires training and $c_{\text{skip}}(\sigma)$, $c_{\text{out}}(\sigma)$, $c_{\text{in}}(\sigma)$, $c_{\text{noise}}(\sigma)$ are standard EDM preconditioning coefficients.

3.4.2 Sampling Process

Sampling is performed in the latent space with $\boldsymbol{Z}_{\text{train}}$ as context. We initialize each entry of $\hat{\boldsymbol{Z}}_{\sigma_{\max}}$ from $\mathcal{N}(0, \sigma_{\max}^2)$ and denoise from $\sigma_{\max}$ to $\sigma_{\min}$. Under the EDM parametrisation, we update:

$$\hat{\boldsymbol{Z}}_{\sigma_{i-1}} = \hat{\boldsymbol{Z}}_{\sigma_i} + (\sigma_{i-1} - \sigma_i)\frac{\hat{\boldsymbol{Z}}_{\sigma_i} - \text{Denoiser}(\hat{\boldsymbol{Z}}_\sigma, \sigma_i; \boldsymbol{Z}_{\text{train}})}{\sigma_i}.$$

Among them, σ_i is a set of preset noise amplitudes from $\sigma_{\max}$ to $\sigma_{\min}$. The generated latent embedding $\hat{\boldsymbol{Z}}$ is obtained after multiple steps of denoising.

3.4.3 In-Context Learning Model

The in-context learning model D_θ in Fig 2 is an 18 layers Transformer and takes the concatenation of the clean embeddings $\boldsymbol{Z}_{\text{train}}$ and noisy embeddings $\boldsymbol{Z}_\sigma$ as input. Specifically, we employ a sample-wise attention mask: clean embeddings attend to each other but cannot attend to noisy embeddings; and each noisy embedding attends only to itself and clean embeddings, but not to other noisy embeddings. This allows the model to leverage the context for guided denoising while keeping the denoising processes of different noisy embeddings independent. For the noise level σ, we apply sinusoidal encoding followed by a shallow MLP that projects it to 192 dimensions, and add it to the noisy query embeddings so that the model is aware of the current noise level.

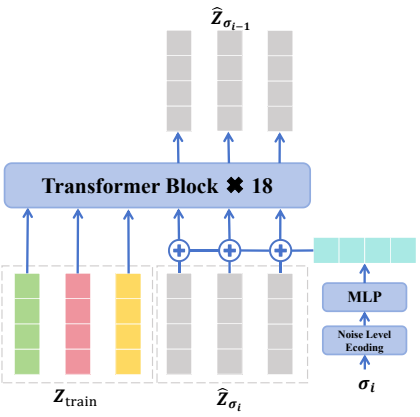


Figure 2: In-context learning model.

3.5 Dataset-Specific MLP Decoder

After noise sampling and multi-step denoising, we obtain the generated latent embeddings $\hat{\boldsymbol{Z}}$, which still needs to be mapped back to the concrete data space. Because different datasets may exhibit distinct feature types, scales, and distributions, a shared decoder is insufficient. We train a lightweight MLP decoder separately for each dataset. This decoder is trained using the original samples $\boldsymbol{X}_{\text{train}}$ and their corresponding latent embeddings $\boldsymbol{Z}_{\text{train}}$. For any feature f, the corresponding MLP decoder maps the latent embedding $\boldsymbol{Z}_{\text{train}}^{(f)}$ to either a scalar value or a probability distribution, and then computes the loss against the true $\boldsymbol{X}_{\text{train}}^{(f)}$. Our optimization objective is as follows:

$$\ell_{\text{dec}} = \sum_f \text{loss}_f\left(\text{MLPDecoder}_f(\boldsymbol{Z}_{\text{train}}), \boldsymbol{X}_{\text{train}}^{(f)}\right).$$

Here, loss_f denotes squared error for numerical features and the cross-entropy loss for categorical features. For the latent embeddings $\hat{\boldsymbol{Z}}$ generated by the diffusion model, the final generated data $\hat{\boldsymbol{X}}$ can be obtained by feeding the corresponding embeddings into the associated decoder. Although this decoder is trained separately for each dataset at inference time, its lightweight architecture incurs only limited additional cost.

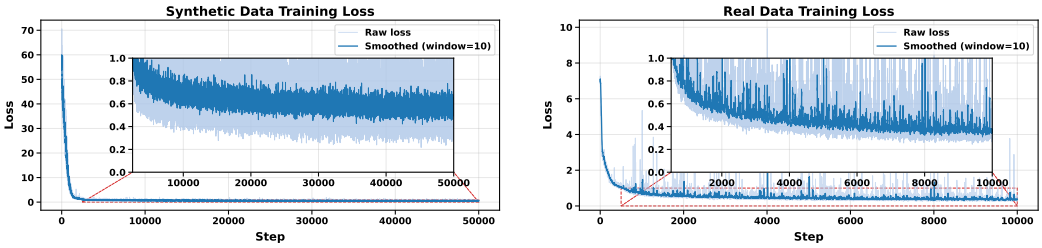


Figure 3: Loss curves during the two-stage training. The light blue curve shows the raw loss, and the dark blue curve shows the loss smoothed by a moving average with a window size of 10.

4 Training Details

4.1 Training Data

Following TabPFN, a general-purpose pretrained model for tabular data, we first train our model on large-scale synthetic data. Specifically, we use the open-source TabICL data generator to produce 5,000 batches of datasets. Each batch contains 512 datasets generated from different SCMs (Pearl, 2009; Poinsot et al., 2024), where each dataset has between 128 and 2,048 samples and between 2 and 50 features. We reuse these 5,000 batches 10 times and randomly shuffle the columns, yielding 50,000 usable training batches in total.

For the second stage, we train on 72 real-world datasets from OpenML-CC18 (Bischl et al., 2019) and OpenML-CTR23 (Fischer et al., 2023), sampling rows and columns within the same ranges as the synthetic data. In each round, we sample one subset per dataset and group all 72 subsets into a single batch. Repeating this 1,000 times yields 1,000 batches, and reusing each batch 10 times gives 10,000 training batches in total.

4.2 Training Process

We first train the model on the aforementioned 50,000 synthetic training batches and then on 10,000 real-data training batches. For each dataset, we uniformly sample a context ratio between 0.5 and 0.8, and use the remaining portion for noising and reconstruction. The entire training process is conducted on a machine equipped with an Intel Xeon Platinum 8336C CPU and 4 NVIDIA RTX 4090 GPUs. It takes about 8 days in total, with the first 7 days using synthetic data and the last day using real data. The loss curve is shown in the figure 3. We use a cosine learning rate schedule with linear warmup. The initial learning rate is set to 2×10^{-4} with a warmup proportion of 0.02, gradient clipping of 1.0.

5 Experiment

This section presents our experimental setup and results. We conduct experiments on 18 real-world datasets, consisting of 7 classification tasks and 11 regression tasks. The generated data are evaluated in terms of statistical fidelity, privacy preservation, and downstream model performance. We will introduce our experimental setup and results step by step.

5.1 Datasets

We use the tabular benchmark TALENT (Ye et al., 2024; Liu et al., 2024), which contains 300 high-quality tabular datasets. We employ a subset provided by them, Tiny Talent, which consists of 45 datasets. After removing datasets with more than 50 features, we obtain 7 classification tasks and 11 regression tasks. Detailed information about the datasets is provided in Appendix A.

To systematically evaluate various generative algorithms under different data budgets, we subsample from each dataset to construct training sets of sizes 16, 32, 64, 128, 256, and 512. These subsets are then used to train the generative models. For each training size, we repeat the experiment 5

Table 1: Relative performance (mean ± std) of different generation methods under various training sizes. For classification, we report $D_{\text{syn}}/D_{\text{real}}$ (gen/real, higher is better); for regression, we report $D_{\text{real}}/D_{\text{syn}}$ (real/gen, higher is better). Standard deviations are shown as subscripts. The best, second-best, and third-best results among generative methods are highlighted. Real is shown as a reference and is not ranked.

Classification (gen/real, higher is better)

Method	16	32	64	128	256	512
Real	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$
SMOTE	$0.9927_{\pm 0.0329}$	$0.9885_{\pm 0.0456}$	$0.9990_{\pm 0.0249}$	$0.9979_{\pm 0.0184}$	$0.9997_{\pm 0.0171}$	$0.9980_{\pm 0.0104}$
TVAE	$1.0035_{\pm 0.0933}$	$0.9761_{\pm 0.1022}$	$0.9502_{\pm 0.0926}$	$0.9368_{\pm 0.0574}$	$0.9310_{\pm 0.0772}$	$0.8988_{\pm 0.0685}$
CTGAN	$0.9918_{\pm 0.1292}$	$0.9611_{\pm 0.1227}$	$0.9231_{\pm 0.1022}$	$0.9019_{\pm 0.1083}$	$0.9162_{\pm 0.0976}$	$0.8842_{\pm 0.1055}$
TabDDPM	$0.9533_{\pm 0.0966}$	$0.9046_{\pm 0.1158}$	$0.9306_{\pm 0.1159}$	$0.9176_{\pm 0.1201}$	$0.9028_{\pm 0.1721}$	$0.8925_{\pm 0.1348}$
ARF	$0.9720_{\pm 0.0648}$	$0.9968_{\pm 0.0697}$	$0.9702_{\pm 0.0510}$	$0.9725_{\pm 0.0450}$	$0.9627_{\pm 0.0273}$	$0.9352_{\pm 0.0396}$
BN	$1.0128_{\pm 0.0710}$	$0.9683_{\pm 0.0428}$	$1.0007_{\pm 0.0747}$	$0.9999_{\pm 0.0653}$	$0.9907_{\pm 0.0663}$	$0.9867_{\pm 0.0621}$
NFlow	$0.9754_{\pm 0.1095}$	$0.9728_{\pm 0.1146}$	$0.9411_{\pm 0.0907}$	$0.9469_{\pm 0.0758}$	$0.8954_{\pm 0.1029}$	$0.8581_{\pm 0.0949}$
TabEBM	$1.0542_{\pm 0.0939}$	$1.0006_{\pm 0.0663}$	$0.9890_{\pm 0.0285}$	$0.9961_{\pm 0.0279}$	$0.9799_{\pm 0.0434}$	$0.9612_{\pm 0.0500}$
TabGift	$1.0729_{\pm 0.1026}$	$1.0313_{\pm 0.0481}$	$1.0213_{\pm 0.0767}$	$1.0030_{\pm 0.0754}$	$0.9923_{\pm 0.0693}$	$0.9526_{\pm 0.0747}$

Regression (real/gen, higher is better)

Method	16	32	64	128	256	512
Real	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$	$1.0000_{\pm 0.0000}$
SMOTE	$0.9762_{\pm 0.1268}$	$1.0451_{\pm 0.1128}$	$1.0153_{\pm 0.2018}$	$0.9618_{\pm 0.1627}$	$0.9437_{\pm 0.1329}$	$0.9192_{\pm 0.1604}$
TVAE	$1.0166_{\pm 0.1737}$	$0.9802_{\pm 0.1310}$	$0.9481_{\pm 0.1645}$	$0.8414_{\pm 0.1437}$	$0.8065_{\pm 0.1530}$	$0.7953_{\pm 0.1801}$
CTGAN	$0.9205_{\pm 0.1799}$	$0.9324_{\pm 0.1235}$	$0.9575_{\pm 0.1921}$	$0.8233_{\pm 0.1447}$	$0.7936_{\pm 0.1422}$	$0.7595_{\pm 0.1230}$
TabDDPM	$0.8655_{\pm 0.1907}$	$0.8642_{\pm 0.2245}$	$0.8477_{\pm 0.2062}$	$0.8476_{\pm 0.2115}$	$0.8133_{\pm 0.1857}$	$0.8105_{\pm 0.1732}$
ARF	$1.0439_{\pm 0.1426}$	$0.9598_{\pm 0.1484}$	$0.9525_{\pm 0.1638}$	$0.8960_{\pm 0.1582}$	$0.8608_{\pm 0.1586}$	$0.8295_{\pm 0.1541}$
BN	$1.1544_{\pm 0.1797}$	$1.0476_{\pm 0.1590}$	$1.0602_{\pm 0.1089}$	$1.0058_{\pm 0.1260}$	$1.0100_{\pm 0.0506}$	$0.9722_{\pm 0.0646}$
NFlow	$0.9805_{\pm 0.1500}$	$0.9167_{\pm 0.1437}$	$0.8799_{\pm 0.1859}$	$0.8039_{\pm 0.1358}$	$0.7703_{\pm 0.1498}$	$0.7580_{\pm 0.1476}$
TabEBM	$1.0836_{\pm 0.1518}$	$1.1073_{\pm 0.1386}$	$1.0870_{\pm 0.1195}$	$1.0297_{\pm 0.0892}$	$1.0303_{\pm 0.0691}$	$0.9772_{\pm 0.0751}$
TabGift	$1.1814_{\pm 0.1658}$	$1.1222_{\pm 0.1925}$	$1.1338_{\pm 0.2140}$	$1.0247_{\pm 0.2000}$	$0.9076_{\pm 0.1703}$	$0.8359_{\pm 0.1992}$

times using 5 different random seeds. We save the generated data produced by each method under each training size for subsequent evaluation and comparison. This protocol allows us to assess how generation quality varies with the amount of available training data.

For baselines that require standardized data, including TabGift, we preprocess each dataset by applying Z-score normalization to continuous features and ordinal encoding to categorical features. This ensures that all methods operate on comparable input representations.

5.2 Baseline Generators And Downstream Predictors

We compare TabGift with existing tabular data generation methods, including: (i) the classical data augmentation method SMOTE (Chawla et al., 2002); (ii) the VAE-based method TVAE (Xu et al., 2019); (iii) the GAN-based method CTGAN (Xu et al., 2019); (iv) the diffusion-based method TabDDPM (Kotelnikov et al., 2023); (v) the tree-based method ARF (Watson et al., 2023); (vi) a Bayesian network (BN)-based method (Ankan & Textor, 2024); (vii) the normalizing flow method NFlow (Durkan et al., 2019); and (viii) TabEBM (Margeloiu et al., 2024). For some of the methods that require hyperparameter tuning, we performed 20 optimization trials. Implementation details can be found in the Appendix B.

We select 6 representative downstream predictors: XGBoost (Chen & Guestrin, 2016), Random Forest (Breiman, 2001), TabPFN (Hollmann et al., 2025), multi-layer perceptrons (Rumelhart et al., 1986), k-nearest neighbors (Cover & Hart, 1967), and Linear Regression or Logistic Regression (Cox, 1958). For some downstream predictors, we perform hyperparameter tuning, while synthetic data is used to train the models.

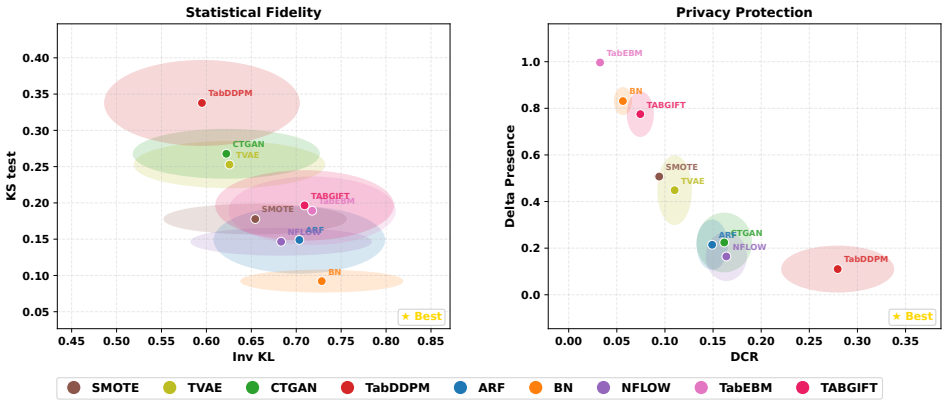


Figure 4: Synthetic generator comparison on statistical fidelity (Inv KL, KS test) and privacy protection (DCR, Delta Presence). Shown for training size 16 across all datasets, mean over 5 seeds.

5.3 Evaluation

5.3.1 Downstream Model Efficacy

In the actual evaluation, downstream models are trained either using only the real training data or using synthetic data generated by different generation methods on top of the training data. We denote the performance of models trained on real data as D_{real}, and the performance of models trained on synthetic data as D_{syn}. We adopt a relative performance metric for evaluation. Specifically, the performance achieved using real data is normalized to 1. For classification tasks, we use the Area Under the Receiver Operating Characteristic curve (AUROC) as the evaluation metric, where higher values indicate better performance. The relative performance is thus computed as $D_{\text{syn}}/D_{\text{real}}$. For regression tasks, we instead use the Root Mean Squared Error (RMSE), where lower values indicate better performance. Accordingly, the relative performance is defined as $D_{\text{real}}/D_{\text{syn}}$. The detailed results are shown in Tab 1.

TabGift consistently outperforms dataset-specific generative methods in the limited-context setting. Dataset-specific generative models are prone to memorizing training samples: with only a handful of examples, the empirical distribution is easily memorized, and the model struggles to distinguish generalizable structure from sample-specific patterns. In contrast, during ICL pretraining, the context is constantly changing and memorizing the context cannot reduce the pretraining loss. This removes the memorization incentive and forces the model to infer transferable distributional priors, enabling high-quality generation from limited context. As the number of training samples increases, the hypothesis spaces of dataset-specific models expand with the data, and their empirical fitting ability gradually strengthens. In contrast, The performance TabGift is mainly limited by the expressiveness of the pretrained prior and the information bottleneck of context compression, so its relative advantage narrows. Nevertheless, even at size 512, TabGift remains in the upper half, indicating that its prior does not fail but that its marginal benefit decreases as the data volume grows.

5.3.2 Statistical Fidelity And Privacy Protection

For statistical fidelity, we estimate the similarity between the training data and the synthetic data using: (i) Column-wise distribution matching: Kullback–Leibler divergence (Csiszár, 1975) is computed for categorical features, Wasserstein distance (Monge, 1781) for numerical features, and the results are inversely mapped to $[0, 1]$ range and then averaged; (ii) Kolmogorov–Smirnov test (Karson, 1968) statistic; For privacy protection, we assess the risk of data leakage using the following two metrics (computed with respect to the sampled training set): (i) Average Distance to Closest Record (Zhao et al., 2021): larger values indicate greater differences between generated and training samples; (ii) δ-presence (van Tilborg & Jajodia, 2011): smaller values indicate a lower probability of identifying training samples from the generated data. We report the results for training size 16 in Fig 4. TabGift performs reasonably well on statistical fidelity but is slightly weaker on privacy preservation. Interestingly, we observe a tension between these two objectives: methods with strong statistical fidelity (e.g., BN, TabEBM, and TabGift) tend to lag behind in privacy

Table 2: Average runtime (seconds, mean) of different generation methods under various training sizes. For each size, we report training time, inference time, and total time.

Method	16			32			64			128			256			512		
	Train	Infer	Total	Train	Infer	Total	Train	Infer	Total	Train	Infer	Total	Train	Infer	Total	Train	Infer	Total
SMOTE	0.000	0.010	0.010	0.000	0.009	0.009	0.000	0.009	0.009	0.000	0.013	0.013	0.000	0.011	0.011	0.000	0.014	0.014
TVAE	17.316	0.161	17.477	13.607	0.168	13.775	17.950	0.148	18.098	17.986	0.197	18.182	33.206	0.130	33.336	54.554	0.167	54.721
CTGAN	30.704	0.128	30.832	32.919	0.129	33.049	50.764	0.113	50.878	55.871	0.101	55.972	69.029	0.095	69.123	94.568	0.097	94.665
TabDDPM	29.110	333.780	362.890	29.493	135.185	164.677	28.566	71.906	100.472	26.288	67.191	93.479	24.295	20.538	44.834	26.733	19.629	46.362
ARF	1.950	3.535	5.486	1.484	3.156	4.640	1.712	3.276	4.988	1.858	2.802	4.660	2.925	3.299	6.224	4.484	2.853	7.337
BN	7.651	0.164	7.814	6.531	0.356	6.887	6.413	0.398	6.811	7.112	0.395	7.508	7.964	0.363	8.327	8.468	0.369	8.837
NFlow	18.243	0.826	19.069	16.338	0.706	17.044	33.076	0.963	34.039	33.815	0.980	34.795	43.249	1.242	44.491	37.970	0.933	38.903
TabEBM	0.000	27.150	27.150	0.000	27.009	27.009	0.000	27.923	27.923	0.000	28.793	28.793	0.000	29.490	29.490	0.000	28.098	28.098
TabGift	8.033	1.976	10.009	8.033	2.005	10.038	8.098	2.021	10.119	9.374	2.180	11.554	7.986	2.421	10.407	7.796	2.908	10.704

protection. A plausible explanation is that better distributional matching often requires preserving fine-grained statistics of the training data, which can bring generated samples closer to real records and thus increase privacy risk. This suggests that statistical fidelity and privacy preservation should be evaluated jointly rather than in isolation.

5.3.3 Running Time Analysis

In tab 2, we generate 1,024 synthetic samples under different training sizes and compare the runtime of all methods on the Adult dataset. SMOTE is by far the fastest (≈0.015s on average), since it only performs interpolation and requires no model training. TabEBM is also training-free, but it relies on multiple calls to TabPFN; its inference time alone (≈28s) already exceeds the total time of TabGift (≈10.5s). Among methods that require training, TabGift is only slower than ARF (≈5.6s) and BN (≈7.7s), while being substantially faster than TVAE (≈25.9s), CTGAN (≈55.8s), TabDDPM (≈135.5s), and NFlow (≈31.4s). The runtime of TabGift is dominated by training the dataset-specific MLP decoder (≈8.2s), and inference accounts for only about 20% of the total time (≈2.3s). This confirms that TabGift achieves a favorable efficiency–quality trade-off, especially in the small-data regime.

6 Limitations

SwiftPFN is pretrained for prediction, so its embeddings are not necessarily optimal for representing a sample in generation, and a dataset-specific MLP decoder is still required. We have explored alternatives such as a universal decoder and jointly training the encoder and decoder as a general tabular VAE, but both yielded inferior results. More systematic efforts are needed to design an encoder–decoder pair that is both transferable and generation-oriented.

Pretraining on synthetic data alone leads to suboptimal generation quality, and adding real data improves performance, suggesting that the synthetic prior is insufficiently diverse or realistic. Future work could improve the quality of synthetic pretraining data, scale up real-data pretraining, and increase per-dataset size and feature dimensionality.

7 Conclusion

TabGift is an efficient foundation model for tabular data generation, designed to address the key limitations of existing generative methods in small-data regimes, including overfitting, poor privacy protection, and high computational cost. Its core idea is to shift the burden of distribution learning from the target dataset to foundation-model pretraining: by pretraining on large-scale heterogeneous tabular data, TabGift acquires transferable structural and distributional priors, enabling high-quality synthetic data generation from only a few context samples. TabGift has three components: a frozen SwiftPFN encoder mapping heterogeneous samples to a 192-d latent space; an 18-layer Transformer denoiser with a sample-wise attention mask that enables context-guided denoising while keeping query denoising independent; and a lightweight dataset-specific MLP decoder mapping latents back to the data space. Training proceeds in two stages: large-scale pretraining on synthetic data (50,000 batches), followed by fine-tuning on 72 real-world datasets (10,000 batches). Experiments on 7 classification and 11 regression tasks show that TabGift consistently achieves the best or near-best generation quality in the limited-context regime (size 16–64), while being substantially more efficient than most baselines.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, including polishing and rephrasing text, and for coding assistance, including debugging and implementing standard routines. We have not used generative AI tools for research ideation, problem formulation, or methodological design. These aspects, including the core idea, method, and experimental design, were conceived and carried out by the authors.

Additionally, we used generative AI tools for literature search and other auxiliary tasks such as summarizing related work and checking terminology. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by the authors, and AI-assisted text was carefully checked and revised. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personal data collection, or any interaction with human participants. All datasets used in our experiments are publicly available and were obtained from standard open benchmarks. We did not collect, release, or redistribute any new datasets.

REPRODUCIBILITY STATEMENT

We have made efforts to ensure the reproducibility of our work. Detailed descriptions of the model architecture, training procedure, and experimental setup are provided in the main paper and the appendix. All datasets used in our experiments are publicly available, and the corresponding preprocessing and sampling procedures are described in the appendix. We intend to release the pretrained model together with the accompanying training and inference code upon publication, which will allow the results reported in this work to be reproduced. The authors are committed to releasing these resources in a timely manner.

REFERENCES

Latent diffusion models with correlation loss for tabular data generation. In *Proceedings of the International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pp. 1849–1853, 2025.

Generative adversarial network-based image and tabular data generation with differential privacy. In *Proceedings of the IEEE Conference*, 2025. IEEE Xplore.

Limix-2m: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Ankur Ankan and Johannes Textor. pgmpy: A python toolkit for bayesian networks. *Journal of Machine Learning Research*, 25(265):1–8, 2024.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *arXiv:1708.03731v2 [stat.ML]*, 2019.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

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.

Jillian M. Clements, Di Xu, Nooshin Yousefi, and Dmitry Efimov. Sequential deep learning for credit risk monitoring with tabular financial data. *CoRR*, 2020.

Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

David R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society, Series B*, 20(2):215–232, 1958.

Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, pp. 146–158, 1975.

Jeffrey Dastin. Amazon scraps secret ai recruiting tool that showed bias against women. In *Ethics of Data and Analytics*, pp. 296–299. Auerbach Publications, 2022.

Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 322–331, 2021.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in Neural Information Processing Systems*, 32, 2019.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. Exaone tabular 1.0: A synthetic-prior foundation model for tabular classification and regression. *arXiv preprint arXiv:2608.25774*, 2026.

Sebastian Felix Fischer, Liana Harutyunyan, Matthias Feurer, and Bernd Bischl. Openml-ctr23 a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023. URL https://openreview.net/forum?id=HebAOoMm94.

Mengchun Gong, Shuang Wang, Lezi Wang, Chao Liu, Jianyang Wang, Qiang Guo, Hao Zheng, Kang Xie, Chenghong Wang, and Zhouguang Hui. Evaluation of privacy risks of patients' data in china: Case study. *JMIR Medical Informatics*, 8(2):e13046, 2020.

Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy-based model and you should treat it like one. In *Proceedings of the International Conference on Learning Representations*, 2020.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-2.5: Advancing the state of the art in tabular foundation models, 2025. URL https://arxiv.org/abs/2511.08667.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-3: Technical report, 2026. URL https://arxiv.org/abs/2605.13986.

Xinyan Han, Yan Lu, Xiaoyu Lin, Yuanyuan Jiang, Yuanrui Wang, Xuanyue Li, Wenchao Zou, and Xingxuan Zhang. Breaking the quality-privacy tradeoff in tabular data generation via in-context learning. *arXiv preprint*, 2026.

Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 1322–1328, 2008.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations 2023*, 2023.

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

Feiran Huang, Zefan Wang, Xiao Huang, Yufeng Qian, Zhetao Li, and Hao Chen. Aligning distillation for cold-start item recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1147–1157, 2023.

Xiangjian Jiang, Mingxuan Liu, Nikola Simidjievski, Tassilo Klein, and Mateja Jamnik. A generative foundation model for heterogeneous tabular data. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Benjamin Jäger, Nick Erickson, Léo Grinsztajn, Felix Birkel, Klemens Flöge, Oscar Key, Kürşat Kaya, Jonas Kübler, Adèle Frankel, Tobias Schröder, Anurag Garg, Jan Hendrik Metzen, David Salinas, Simon Bing, Kristina Collins, Tuana Çelik, Vahid Balazadeh, Lydia Sidhoum, Tomás Pereda, Brendan Roof, Andrej Tschalzev, Siyuan Guo, Philipp Singer, Lennart Purucker, Jake Robertson, Marie Salmon, Philipp Jund, Jerry Chen, Diana Kriuchkova, Arthur Cahu, Eliott Kalfon, Adrian Hayler, Georg Grab, Vitor Monteiro, Lilly Wehrhahn, Dominik Safaric, Clara Cornu, Alan Arazi, Rylee Grace, Simone Alessi, Mihir Manium, Bernhard Schölkopf, Yann LeCun, Madelon Hulsebos, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. Tabpfn-3.5: Technical report, 2026. URL `https://arxiv.org/abs/2609.17895`.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577, 2022.

Marvin Karson. Review of "handbook of methods of applied statistics", 1968.

Weihao Kong and Abhimanyu Das. Introducing tabfm: A zero-shot foundation model for tabular data. Google Research Blog, June 2026. URL `https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/`.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 17564–17579, 2023.

Guillaume Lemaitre, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

Xiaofeng Lin, Chenheng Xu, Matthew Yang, and Guang Cheng. Ctsyn: A foundation model for cross tabular data generation. In *Proceedings of the International Conference on Learning Representations*, 2025.

Xiaofeng Lin, Seungbae Kim, Zhuoya Li, Zachary DeSoto, Charles Fleming, and Guang Cheng. Retabsyn: Realistic tabular data synthesis via reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2026.

Si-Yang Liu and Han-Jia Ye. Swiftpfn: Revisiting row-wise attention–only tabular foundation models with adaptive early exit. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. Spotlight.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *arXiv preprint arXiv:2407.04057*, 2024.

Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. Tabpfgen: Tabular data generation with tabpfn. In *NeurIPS 2023 Workshop on Table Representation Learning*, 2023.

Andrei Margeloiu, Xiangjian Jiang, Nikola Simidjievski, and Mateja Jamnik. Tabebm: A tabular data augmentation method with distinct class-specific energy-based models. *Advances in Neural Information Processing Systems*, 37:72094–72144, 2024.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6):1–35, 2021.

Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Mémoires de l'Académie Royale des Sciences*, pp. 666–704, 1781.

Dae-Young Park and In-Young Ko. Tabular generative model with graph-theoretical methodologies toward effective financial fraud detection. *ACM Transactions on Knowledge Discovery from Data*, 2026. DOI: 10.1145/3819586.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Audrey Poinsot, Alessandro Leite, Nicolas Chesneau, Michele Sebag, and Marc Schoenauer. Learning structural causal models through deep generative models: Methods, guarantees, and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24), Survey Track*, 2024.

Zhaozhi Qian, Rob Davis, and Mihaela van der Schaar. Synthcity: A benchmark framework for diverse use cases of tabular synthetic data. *Advances in Neural Information Processing Systems*, 36:3173–3188, 2023.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50817–50847. PMLR, 2025.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Zijian Shen, Taijie Chen, Bin Zhou, Ziyang Jiang, and Jintao Ke. Lab-tab: Llm-augmented bayesian network adaptation for few-shot tabular generation. *arXiv preprint*, 2026.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. Tabdiff: A multi-modal diffusion model for tabular data generation. *CoRR*, 2024.

Yefan Tao, Xiyuan Zhang, Xinyi Liu, Boran Han, Danielle Maddix, Haoyang Fang, Zhen Han, Jiading Gai, Xuanqing Liu, Michael Bohlke-Schneider, Yuyang Wang, Gerald Friedland, Kevan Mah, Chris Lee, and Chris Kong. Mitra-v2 technical report. *arXiv preprint arXiv:2609.04540*, 2026.

Henk C. A. van Tilborg and Sushil Jajodia. *Encyclopedia of Cryptography and Security*. Springer, 2011.

Penghui Wang, Wei Liu, Hong Wang, Cheng-Yue Huang, Yu-Xi Sun, Zi-Rui Wang, Hong-Ming Huang, Quan Wang, Chun-Xiao Liu, Er-Li Meng, and Bin Wang. Xiaomi-tabldm: A tabular foundation model technical report. *arXiv preprint arXiv:2609.03880*, 2026a.

Yanbo Wang, Jiaxuan You, Chuan Shi, and Muhan Zhang. Relational in-context learning via synthetic pre-training with structural prior. In *Proceedings of the International Conference on Machine Learning*, 2026b.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, pp. 5357–5375, 2023.

Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *CoRR*, 2021.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. *Advances in Neural Information Processing Systems*, 32, 2019.

Qian Yang, Jianyi Zhang, Weituo Hao, Gregory P. Spell, and Lawrence Carin. Flop: Federated learning on medical datasets using partial networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3845–3853, 2021.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024.

Jinsung Yoon, Lydia N. Drumright, and Mihaela van der Schaar. Anonymization through data synthesis using generative adversarial networks. *IEEE Journal of Biomedical and Health Informatics*, 24(8):2378–2388, 2020.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *Proceedings of the Twelfth International Conference on Learning Representations*, 2024.

Honglei Zhang, Fangyuan Luo, Jun Wu, Xiangnan He, and Yidong Li. Lightfr: Lightweight federated recommendation with privacy-preserving matrix factorization. *ACM Transactions on Information Systems*, 41(4):1–28, 2023.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations*, 2018.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. CTAB-GAN: Effective table data synthesizing. In *Proceedings of the Asian Conference on Machine Learning*, pp. 97–112, 2021.

Yuli Zheng, Zhenyu Wu, Ye Yuan, Tianlong Chen, and Zhangyang Wang. Pcal: A privacy-preserving intelligent credit risk modeling framework based on adversarial learning. *CoRR*, 2020.

Chaoyi Zhu, Jiayi Tang, Juan F. Pérez, Marten van Dijk, and Lydia Y. Chen. Dp-tldm: Differentially private tabular latent diffusion model. In *Proceedings of the International Conference on Availability, Reliability and Security*, 2025.

APPENDIX

A DATASETS

A.1 OPENML

This set of datasets comes from the OpenML-CC18 (Bischl et al., 2019) classification benchmark and the OpenML-CTR23 (Fischer et al., 2023) regression benchmark. OpenML-CC18 is a curated collection of 72 classification datasets, while OpenML-CTR23 is a curated collection of 36 regression datasets. Both are widely used in tabular learning research for standardized and reproducible evaluation. In our experiments, we select 72 datasets from these two benchmarks, covering both classification and regression tasks with diverse feature types, sample sizes, and dimensionalities. Detailed statistics of the selected datasets are provided in Tab 3 and Tab 4.

Table 3: Statistics of the OpenML regression datasets.

Dataset	#Features	#Samples	#Cat / #Num
abalone	8	4,177	1 / 7
airfoil_self_noise	5	1,503	0 / 5
auction_verification	7	2,043	6 / 1
concrete_compressive_strength	8	1,030	0 / 8
physiochemical_protein	9	45,730	0 / 9
superconductivity	81	21,263	2 / 79
geographical_origin_of_music	116	1,059	0 / 116
naval_propulsion_plant	14	11,934	1 / 13
white_wine	11	4,898	0 / 11
grid_stability	12	10,000	0 / 12
video_transcoding	18	68,784	2 / 16
wave_energy	48	72,000	0 / 48
sarcos	21	48,933	0 / 21
california_housing	8	20,640	0 / 8
cpu_activity	21	8,192	0 / 21
diamonds	9	53,940	3 / 6
kin8nm	8	8,192	0 / 8
pumadyn32nh	32	8,192	0 / 32
socmob	5	1,156	4 / 1
kings_county	21	21,613	7 / 14
brazilian_houses	9	10,692	6 / 3
fps_benchmark	43	24,624	19 / 24
health_insurance	11	22,272	9 / 2
Moneyball	14	1,232	4 / 10
energy_efficiency	8	768	2 / 6
student_performance_por	30	649	29 / 1
QSAR_fish_toxicity	6	908	2 / 4
cars	17	804	16 / 1
space_ga	6	3,107	0 / 6

Table 4: Statistics of the OpenML classification datasets.

Dataset	#Classes	#Features	#Samples	#Cat / #Num	Dataset	#Classes	#Features	#Samples	#Cat / #Num
kr-vs-kp	2	36	3,196	36 / 0	pc1	2	21	1,109	0 / 21
letter	26	16	20,000	16 / 0	adult	2	14	48,842	9 / 5
balance-scale	3	4	625	0 / 4	Bioresponse	2	1,776	3,751	834 / 942
mfeat-factors	10	216	2,000	22 / 194	wdbc	2	30	569	0 / 30
mfeat-fourier	10	76	2,000	0 / 76	phoneme	2	5	5,404	0 / 5
breast-w	2	9	699	0 / 9	qsar-biodeg	2	41	1,055	22 / 19
mfeat-karhunen	10	64	2,000	0 / 64	wall-robot-navigation	4	24	5,456	0 / 24
mfeat-morphological	10	6	2,000	0 / 6	semeion	10	256	1,593	256 / 0
mfeat-zernike	10	47	2,000	0 / 47	ilpd	2	10	583	1 / 9
cmc	3	9	1,473	8 / 1	ozone-level-8hr	2	72	2,534	0 / 72
optdigits	10	64	5,620	0 / 64	cnae-9	9	856	1,080	856 / 0
credit-approval	2	15	690	9 / 6	first-order-theorem-proving	6	51	6,118	0 / 51
credit-g	2	20	1,000	17 / 3	banknote-authentication	2	4	1,372	0 / 4
pendigits	10	16	10,992	0 / 16	blood-transfusion-service-center	2	4	748	0 / 4
diabetes	2	8	768	0 / 8	PhishingWebsites	2	30	11,055	30 / 0
spambase	2	57	4,601	0 / 57	cylinder-bands	2	37	540	16 / 21
splice	3	60	3,190	60 / 0	bank-marketing	2	16	45,211	9 / 7
vehicle	4	18	846	1 / 17	GesturePhaseSegmentationProcessed	5	32	9,873	0 / 32
electricity	2	8	45,312	1 / 7	har	6	561	10,299	0 / 561
satimage	6	36	6,430	0 / 36	dresses-sales	2	12	500	11 / 1
eucalyptus	5	19	736	6 / 13	texture	11	40	5,500	0 / 40
sick	2	29	3,772	22 / 7	connect-4	3	42	67,557	42 / 0
vowel	11	12	990	2 / 10	MiceProtein	8	77	1,080	0 / 77
isolet	26	617	7,797	0 / 617	steel-plates-fault	7	27	1,941	2 / 25
analcatdata_authorship	4	70	841	29 / 41	climate-model-simulation-crashes	2	18	540	0 / 18
analcatdata_dmft	5	4	797	4 / 0	wilt	2	5	4,839	0 / 5
mnist_784	10	784	70,000	150 / 634	car	4	6	1,728	6 / 0
pc4	2	37	1,458	3 / 34	segment	7	16	2,310	0 / 16
pc3	2	37	1,563	2 / 35	Fashion-MNIST	10	784	70,000	3 / 781
jm1	2	21	10,885	0 / 21	jungle_chess_2pcs_raw_endgame_complete	3	6	44,819	6 / 0
kc2	2	21	522	1 / 20	numerai28.6	2	21	96,320	0 / 21
kc1	2	21	2,109	1 / 20	Devnagari-Script	46	1,024	92,000	240 / 784
					CIFAR_10	10	3,072	60,000	0 / 3,072
					Internet-Advertisements	2	1,558	3,279	1,555 / 3
					dna	3	180	3,186	180 / 0
					churn	2	20	5,000	4 / 16

A.2 TALENT

TALENT (Liu et al., 2024) is a comprehensive benchmark and toolbox for tabular data learning, providing a large collection of high-quality tabular datasets together with standardized pipelines for data preprocessing, model training, and evaluation. It includes 300 tabular datasets spanning classification and regression tasks, and provides a smaller curated subset, TALENT-tiny, for efficient and reproducible evaluation. In our experiments, we select 18 datasets from TALENT-tiny as our test set, covering both classification and regression tasks. Detailed statistics of the selected datasets are provided in Tab 4 and Tab 3.

Table 5: Statistics of the TALENT classification datasets.

Dataset	#Rows	#Features	#Cat	#Num	#Classes
FOREX_audchf-day-High	1,833	10	0	10	2
FOREX_audsgd-hour-High	43,825	10	0	10	2
FOREX_cadjpy-hour-High	43,825	10	0	10	2
Gender_Gap_in_Spanish_WP	4,746	13	0	13	3
adult	45,222	14	8	6	2
moons	512	2	0	2	2
pc3	1,563	37	1	36	2
taiwanese_bankruptcy_prediction	6,819	95	2	93	2
website_phishing	1,353	9	9	0	3

B BASELINES

Below is a detailed introduction and implementation of the baseline methods used in this paper:

- **SMOTE** (Chawla et al., 2002)

  Introduction: Synthetic Minority Over-sampling Technique generates new samples for minority classes via interpolation. It helps balance imbalanced datasets and improves classifier performance on underrepresented classes.

Table 6: Statistics of the TALENT regression datasets.

Dataset	#Rows	#Features	#Cat	#Num	#Unique Targets
Ailerons	13,750	40	12	28	35
CookbookReviews	18,182	7	1	6	6
IEEE80211aa-GATS	4,046	27	2	25	3,407
Large-scale_Wave_Energy_Farm_Sydney_49	17,964	99	0	99	4,816
Superconductivty	21,197	81	2	79	3,007
VulNoneVul	5,692	16	0	16	2
archive2	1,143	12	1	11	1,143
bank8FM	8,192	8	1	7	7,913
communities_and_crime	1,994	102	2	100	98
compass_reg	16,644	17	8	9	2
fried	40,768	10	0	10	17,396
house_16H_reg	22,784	16	0	16	2,045
kin8nm	8,192	8	0	8	8,191

Implementation: Use the open-source implementation of SMOTE from the python library Imbalanced-learn (Lemaitre et al., 2017). `https://imbalanced-learn.org/stable/`.

Hyperparameter Space:

k: integer, choices $\{1, 3, 5\}$

- **TVAE** (Xu et al., 2019):

  Introduction: A Variational Autoencoder designed for tabular data that captures feature dependencies in a latent space. It generates realistic synthetic samples while preserving correlations among columns.

  Implementation: Use the open-source implementation in the python library Synthcity (Qian et al., 2023). `https://github.com/vanderschaarlab/synthcity/tree/main`.

  Hyperparameter Space:

  n_iter: integer, choices $\{100, 200, 300, 400, 500\}$
  lr: float, choices $\{0.0001, 0.0002, 0.001\}$
  decoder_n_layers_hidden: integer, range $[1, 5]$
  weight_decay: float, choices $\{0.0001, 0.001\}$
  batch_size: integer, choices $\{64, 128, 256, 512\}$
  n_units_embedding: integer, range $[50, 500]$ step 50
  decoder_n_units_hidden: integer, range $[50, 500]$ step 50
  decoder_nonlin: categorical, choices {elu, leaky_relu, relu, tanh}
  decoder_dropout: float, range $[0.0, 0.2]$
  encoder_n_layers_hidden: integer, range $[1, 5]$
  encoder_n_units_hidden: integer, range $[50, 500]$ step 50
  encoder_nonlin: categorical, choices {elu, leaky_relu, relu, tanh}
  encoder_dropout: float, range $[0.0, 0.2]$

- **CTGAN** (Xu et al., 2019):

  Introduction: A GAN-based model for tabular data that uses conditional sampling to handle discrete columns. It effectively models mixed-type tables and generates high-quality synthetic data.

  Implementation: Use the open-source implementation in the python library Synthcity.

  Hyperparameter Space:

  generator_n_layers_hidden: integer, range $[1, 4]$
  generator_n_units_hidden: integer, range $[50, 150]$, step 50
  generator_nonlin: categorical, choices {elu, leaky_relu, relu, tanh}
  n_iter: integer, range $[100, 1000]$, step 100

generator_dropout: float, range $[0.0, 0.2]$
discriminator_n_layers_hidden: integer, range $[1, 4]$
discriminator_n_units_hidden: integer, range $[50, 150]$, step 50
discriminator_nonlin: categorical, choices {elu, leaky_relu, relu, tanh}
discriminator_n_iter: integer, range $[1, 5]$
discriminator_dropout: float, range $[0.0, 0.2]$
lr: float, choices $\{0.0001, 0.0002, 0.001\}$
weight_decay: float, choices $\{0.0001, 0.001\}$
batch_size: integer, choices $\{100, 200, 500\}$
encoder_max_clusters: integer, range $[2, 20]$

- **TabDDPM** (Kotelnikov et al., 2023):

  Introduction: A diffusion-based generative model that denoises samples in data or latent space. It gradually transforms noise into realistic tabular samples using a learned denoising process.

  Implementation: Use the open-source implementation in the python library Synthcity.

  Hyperparameter Space:

  lr: log-uniform, range $[10^{-5}, 0.1]$
  batch_size: integer log-uniform, range $[256, 4096]$
  num_timesteps: integer, range $[10, 1000]$
  n_iter: integer log-uniform, range $[1000, 10000]$

- **ARF** (Watson et al., 2023):

  Implementation: Use the open-source implementation in the python library Synthcity.

  Introduction: Adversarial Random Forests is a non-parametric method for density estimation and synthetic data generation. It can model intricate feature interactions without assuming a parametric form.

  Hyperparameter Space:

  num_trees: integer, range $[10, 100]$, step 10
  delta: float, range $[0.0, 0.5]$
  max_iters: integer, range $[1, 5]$
  early_stop: categorical, choices {False, True}
  min_node_size: integer, range $[2, 20]$, step 2

- **BN** (Ankan & Textor, 2024):

  Introduction: Bayesian Network-based generator models variable dependencies via a directed acyclic graph. New samples are generated by sampling from the joint probability distribution encoded by the network.

  Implementation: Use the open-source implementation in the python library Synthcity.

  Hyperparameter Space:

  struct_learning_search_method: categorical, choices {hillclimb, pc, tree_search}
  struct_learning_score: categorical, choices {bdeu, bds, bic, k2}

- **NFlow** (Durkan et al., 2019):

  Introduction: Neural Flow-based generator leverages normalizing flows to model complex joint distributions. It transforms a simple base distribution into the target data distribution through a sequence of invertible transformations, enabling efficient sampling and exact likelihood estimation.

  Implementation: Use the open-source implementation in the Python library Synthcity.

  Hyperparameter Space:

  n_iter: integer, range $[100, 5000]$, step 100
  n_layers_hidden: integer, range $[1, 10]$
  n_units_hidden: integer, range $[10, 100]$
  batch_size: categorical, choices $\{32, 64, 128, 256, 512\}$
  dropout: float, range $[0.0, 0.2]$

batch_norm: categorical, choices {False, True}
lr: categorical, choices $\{0.0001, 0.0002, 0.001\}$
linear_transform_type: categorical, choices {lu, permutation, svd}
base_transform_type: categorical, choices {affine-autoregressive, affine-coupling, quadratic-autoregressive, quadratic-coupling, rq-autoregressive, rq-coupling}

- **TabEBM** (Margeloiu et al., 2024):

  Implementation: Use the open-source implementation in the python library TabEBM. `https://github.com/andreimargeloiu/TabEBM`.

  Introduction: Combines TabPFN with energy-based models to synthesize high-quality tabular data. This approach balances flexibility and accuracy by leveraging energy-based likelihood estimation.

C Downstream model

Below is a detailed introduction and implementation of the downstream model used in this paper:

- **XGBoost** (Chen & Guestrin, 2016)

  Introduction: A scalable and efficient gradient boosting framework for supervised learning. It constructs an ensemble of decision trees to optimize prediction performance while controlling overfitting.

  Implementation: Use the open-source implementation in the python library xgboost. `https://xgboost.ai/`

  Hyperparameter Space:
  alpha: float, $[10^{-8}, 100]$ (log scale)
  colsample_bylevel: float, $[0.5, 1.0]$
  colsample_bytree: float, $[0.5, 1.0]$
  gamma: float, $[10^{-8}, 100]$ (log scale)
  lambda: float, $[10^{-8}, 100]$ (log scale)
  learning_rate: float, $[10^{-4}, 0.3]$ (log scale)
  max_depth: integer, $[3, 10]$
  min_child_weight: float, $[10^{-2}, 100]$ (log scale)
  subsample: float, $[0.5, 1.0]$

- **Random Forest** (Breiman, 2001)

  Introduction: An ensemble method that builds multiple decision trees and aggregates their predictions. It is robust to overfitting and can capture complex feature interactions.

  Implementation: Use the open-source implementation in the python library scikit-learn (Pedregosa et al., 2011). `https://scikit-learn.org/stable/index.html`

  Hyperparameter Space:
  min_samples_split: integer, $[2, 10]$ **min_samples_leaf**: integer, $[1, 10]$

- **TabPFN** (Hollmann et al., 2025)

  Introduction: A tabular foundation model that performs in-context learning for classification and regression tasks. It can estimate predictive distributions without explicit training on the downstream dataset.

  Implementation: Use the open-source implementation in the python library tabpfn. `https://github.com/PriorLabs/TabPFN`.

- **Linear Regression or Logistic Regression** (Cox, 1958)

  Introduction: Classical parametric models for regression and classification. Linear regression models continuous outcomes, while logistic regression models binary outcomes using a sigmoid function.

  Implementation: Use the open-source implementation in the python library scikit-learn.

- **K-Nearest Neighbors (KNN)** (Cover & Hart, 1967)

  Introduction: A non-parametric method for classification and regression that predicts the output of a sample based on the majority label (for classification) or average value (for regression) of its k nearest neighbors in the feature space.

  Implementation: Use the open-source implementation in the python library scikit-learn. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`

  Hyperparameters:

  n_neighbors: integer, range $[1, 20]$

  weights: categorical, choices $\{\text{uniform}, \text{distance}\}$

  p: integer, range $[1, 3]$

- **Multi-Layer Perceptron (MLP)** (Rumelhart et al., 1986)

  Introduction: A feedforward neural network with one or more hidden layers that can model complex non-linear relationships for regression or classification tasks. Uses back-propagation to learn network weights.

  Implementation: Use the open-source implementation in the python library scikit-learn.

  Hyperparameters:

  hidden_dim: integer, range $[64, 512]$

  num_layers: integer, range $[1, 3]$

  dropout: float, range $[0.0, 0.5]$

  lr: float, range $[1e-4, 1e-2]$ (log scale)

  weight_decay: float, range $[1e-6, 1e-2]$ (log scale)

  batch_size: integer, choices $\{32, 64, 128\}$

  num_epochs: integer, range $[50, 500]$