XGB-DDPM: NOISE-REFRESHED GRADIENT BOOSTING FOR SCALABLE DIFFUSION ON THE CPU

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a method to optimize stochastic denoising objectives with Gradient-Boosted Decision Trees (GBDTs) by refreshing noise realizations across boosting rounds. Previous efforts relied on expanding the training data over many noise realizations, incurring extreme memory and runtime costs, especially for already-large datasets. We implement this method in an XGBoost fork and use it to construct XGB-DDPM: a DDPM-style tabular diffusion model combining Gaussian and multinomial diffusion for numerical and categorical features, which are denoised by XGBoost regressors and classifiers. Here, XGBoost leverages the round-wise noise refresh to estimate the diffusion loss. We show that XGB-DDPM can be trained more efficiently and scales to much larger datasets than previous GBDT-based diffusion models which expand the training data to estimate the diffusion loss. Additionally, we show that XGB-DDPM achieves similar fidelity to neural tabular diffusion models (e.g., TabDDPM, TabSyn, TabDiff, and CDTD) without increasing memorization or membership disclosure risk. XGB-DDPM is a good alternative for practitioners who want to train diffusion models on potentially large data but do not have access to GPUs.

1 INTRODUCTION

Recent years have seen various advances to neural tabular diffusion models to better adapt them to the task of mixed-type tabular synthesis (Kotelnikov et al., 2023; Zhang et al., 2024; Mueller et al., 2025; Shi et al., 2025; Mueller et al., 2026). Despite these advances, neural diffusion models retain several weaknesses for tabular data: they require expensive and often inaccessible hardware for training (GPUs) (Lehdonvirta et al., 2024), elaborate hyperparameter tuning, and proper alignment of numerical and categorical features to the overall loss.

Instead of neural networks, a promising solution is to use Gradient-Boosted Decision Trees (GBDT) as denoiser in diffusion models. GBDTs are known to have suitable inductive biases for mixed-type tabular data, are relatively robust to hyperparameters, and can be efficiently parallelized on CPUs (Grinsztajn et al., 2022). ForestDiffusion first showed that GBDT-based diffusion models can be trained effectively and may even outperform neural diffusion models (Jolicoeur-Martineau et al., 2024). Likewise, Treeffuser showed that GBDTs can be used for probabilistic prediction using conditional diffusion (Beltran Velez et al., 2024).

Estimating the diffusion loss in GBDT-based diffusion models. Effectively estimating diffusion loss requires marginalizing over various noise realizations per observation at each timestep. Neural networks inherently handle this through mini-batch training. Since GBDTs do not use mini-batch training, previous efforts in GBDT-based diffusion models proposed to handle this by duplicating the dataset many times (typically 50-100) (Jolicoeur-Martineau et al., 2024; Beltran Velez et al., 2024). This effectively estimates the diffusion loss, but it can incur huge memory and runtime costs. As an example, the `libras` dataset ($n = 288, d = 90$) required > 150GiB memory in the original ForestDiffusion implementation (Cresswell & Kim, 2024).

As an alternative, we propose to update the noise realization at each boosting round to estimate the diffusion loss. This means we draw a fresh noise sample using a different seed, and reassign the histogram bins. Although this slows down the sequential boosting process, it omits the need for

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.

dataset duplication; therefore it does not incur additional memory, and trees are trained on much less data. We implement this within an XGBoost fork[1].

XGB-DDPM: a DDPM with noise-refreshed GBDT backbone. We showcase the method's effectiveness in a new XGBoost-based diffusion model: XGB-DDPM. This is a Denoising Diffusion Probabilistic Model (DDPM) (Ho et al., 2020) using XGBoost denoisers with round-wise updates of the noise realizations. Here, we use Gaussian diffusion for numerical features which are denoised by XGBoost regressors, and multinomial diffusion for categorical features which are denoised by XGBoost classifiers. Feature-wise diffusion processes are especially effective here, since separate XGBoost models are trained for each feature, such that there is no need to align numerical and categorical losses as would be the case for neural diffusion models (Kotelnikov et al., 2023; Mueller et al., 2025).

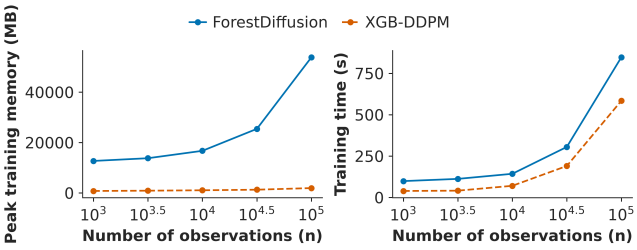

Figure 1: XGB-DDPM avoids the steeply increasing memory burden as the number of observations grows.

Figure 1 shows that our noise update strategy yields much lower memory burden and training time for increasing observations compared to ForestDiffusion's dataset duplication. Here, we already use the implementation from Cresswell & Kim (2024), which is a more memory- and runtime-efficient implementation of ForestDiffusion. In the rest of this work, we show that XGB-DDPM performs on par with state-of-the-art neural tabular diffusion models such as TabDDPM (Kotelnikov et al., 2023), TabSyn (Zhang et al., 2024), TabDiff (Shi et al., 2025), and CDTD (Mueller et al., 2025) in terms of generative quality, while training efficiently for large datasets on the CPU.

2 Related Work

The late 2010s saw an increasing interest in deep generative models for mixed-type tabular data, initially focusing mostly on GANs and VAEs (Xu et al., 2019; Zhao et al., 2021). After the success of neural diffusion models in other domains, various styles were adapted for tabular data as well, e.g., DDPM (Kotelnikov et al., 2023), score-matching (Kim et al., 2023), latent diffusion (Zhang et al., 2024), combinations of score-matching with score-interpolation (Mueller et al., 2025) and masked diffusion (Shi et al., 2025), cascaded diffusion (Mueller et al., 2026), and flow-matching models (Guzmán-Cordero et al., 2025).

Tree-based models are often considered strong methods for supervised tasks in tabular data due to their efficiency and suitable inductive biases (Grinsztajn et al., 2022). This has led to an interest in tree-based generative models for tabular data as well (Watson et al., 2023; McCarter, 2024; Bravo, 2025). Of particular interest for this work, Jolicoeur-Martineau et al. (2024) proposed using GBDTs as backbone in score- and flow-matching models (ForestDiffusion and ForestFlow, respectively). To effectively marginalize over different noise realizations per timestep, their method required duplicating the dataset 50-100 times, leading to severe runtime and memory inefficiencies. Additionally, they use a joint numerical diffusion process for numerical and one-hot encoded categorical features, which is often not suitable for categorical data. These issues did not surface in the originally included benchmark since the datasets had relatively few observations, and were often purely numerical after conditioning on the target. They did, however, lead to ForestDiffusion's omission in many recent works on tabular generative models (Mueller et al., 2025; Shi et al., 2025; Mueller et al., 2026).

[1] An anonymized version is provided in the supplementary material.

3 METHODS

Let $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{n}$ denote a mixed-type tabular dataset of i.i.d. samples $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_d^{(i)})$ drawn from an unknown distribution $p(\mathbf{x})$. The goal of mixed-type tabular synthesis is to learn a generative distribution $p_\theta(\mathbf{x})$ that approximates $p(\mathbf{x})$.

Diffusion models. Diffusion models learn a generative distribution by learning to reverse a forward Markov process $q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$ (Ho et al., 2020). Here, $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is part of a fixed Markov chain that adds noise to the data. The reverse process defines a joint distribution over $\mathcal{X}$:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{1}$$

with a corresponding denoising objective of the general form

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\xi}} \left[\ell \left(f_\theta(\mathbf{x}_t, t), \mathbf{y}(\mathbf{x}_0, \boldsymbol{\xi}, t) \right) \right], \qquad \mathbf{x}_t = g(\mathbf{x}_0, \boldsymbol{\xi}, t), \tag{2}$$

where $\boldsymbol{\xi}$ denotes the stochasticity of the forward process, g generates the corresponding noised observation, and $\mathbf{y}$ denotes the prediction target determined by the chosen diffusion parameterization. Most often, θ is learned by a neural network. However, it can also be learned by XGBoost models (Jolicoeur-Martineau et al., 2024). Specifically, each feature is denoised by a separate XGBoost model. We also train separate models for each timestep and (categorical) target label, following ForestDiffusion (Jolicoeur-Martineau et al., 2024).

ForestDiffusion implements an XGBoost-based variance-preserving score-matching diffusion model and an XGBoost-based conditional flow-matching model (Jolicoeur-Martineau et al., 2024). We choose a DDPM specification instead, as this can be more easily reconciled with previous categorical diffusion paradigms (Hoogeboom et al., 2021), which we wish to use for categorical features. However, we note that these model specifications are in fact very similar, up to the choice of time-discretization, (non-)deterministic sampler, and objective specification (Gao et al., 2024). Furthermore, the advantage that continuous-time score- and flow-matching models may have in sampling flexibility is largely foregone in ForestDiffusion by iterating over timestep-wise XGBoost models. These observations led us to consider DDPM-style diffusion specifications instead of score- or flow-matching as in ForestDiffusion.

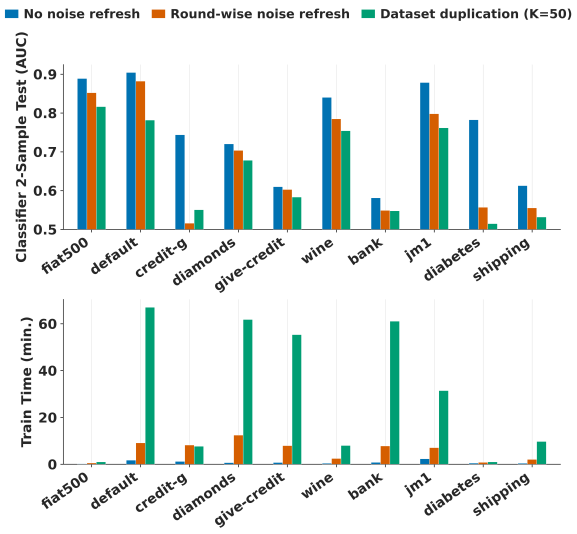

Figure 2: **Refreshing the noise realization over boosting rounds is an efficient alternative to duplicating the dataset.** In particular, it enhances multivariate fidelity at reasonable training cost.

Estimating the expectation in diffusion loss. Approximating the expectation over diffusion noise requires integrating over multiple noise realizations. Previous GBDT-based diffusion models approximate this expectation by duplicating the training set many times, such that each observation is associated with multiple i.i.d. noise realizations (Jolicoeur-Martineau et al., 2024; Beltran Velez et al., 2024). Instead, we sample a fresh noise realization at each boosting round within XGBoost, thereby updating the noised training observations and their histogram bin-assignments across rounds. Figure 2 shows that refreshing the noise realization over boosting rounds improves the learned generative distribution – as measured by the AUC of an XGBoost-based Classifier 2-Sample Test (C2ST) (Friedman, 2004; Lopez-Paz & Oquab, 2017) – at reasonable training time.

Let $F_{b-1}^{j,t}$ denote the boosted model before round b, for feature j and timestep t. At each boosting round, we independently sample $\boldsymbol{\xi}_b$ and construct

$$\mathbf{x}_t^{(i,b)} = g\left(\mathbf{x}_0^{(i)}, \boldsymbol{\xi}_b^{(i)}, t\right), \tag{3}$$

together with the corresponding target $y_t^{j,(i,b)}$. The boosting update is therefore computed from the stochastic empirical objective

$$\widehat{\mathcal{L}}_b^{j,t} = \frac{1}{n}\sum_{i=1}^{n} \ell_j\left(F_{b-1}^{j,t}\left(\mathbf{x}_t^{(i,b)}\right), y_t^{j,(i,b)}\right), \qquad \boldsymbol{\xi}_b^{(i)} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{\xi}). \tag{4}$$

Its conditional expectation, given the current boosted model, is

$$\mathbb{E}_{\boldsymbol{\xi}_b}\left[\widehat{\mathcal{L}}_b^{j,t} \,\middle|\, F_{b-1}^{j,t}\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{\xi}}\left[\ell_j\left(F_{b-1}^{j,t}\left(g\left(\mathbf{x}_0^{(i)}, \boldsymbol{\xi}, t\right)\right), y^j\left(\mathbf{x}_0^{(i)}, \boldsymbol{\xi}, t\right)\right)\right]. \tag{5}$$

Thus, conditional on the current boosted model, each boosting round uses an unbiased Monte Carlo estimate of the expectation over diffusion noise. This also provides a direct connection to stochastic gradient boosting. Gradient boosting can be viewed as stage-wise gradient descent in function space (Friedman, 2001), while stochastic gradient boosting uses a randomized estimate of the objective at each boosting round (Friedman, 2002). In our case, the randomness arises from independently re-sampling the diffusion noise, instead of, e.g., subsampling observations. Repeated boosting rounds therefore optimize the expected diffusion objective across different noise realizations, without explicitly storing multiple noisy copies of every observation. This procedure requires recomputing the noised inputs and their histogram bin-assignments before each boosting round, which we implement within an XGBoost fork[1].

Boosting requirements under noise refreshing. The number of boosting rounds B and learning rate η jointly control the degree to which boosting averages over diffusion noise. Since each boosting round observes only one Monte Carlo realization of the diffusion objective, each denoiser should be trained for large enough B such that many independent noise realizations affect the fitted model. Additionally, η should be small enough such that noise realizations from earlier rounds do not disproportionately affect the full model. Generally, this means that B should be set as high as possible while still yielding reasonable training times given available resources and the dataset. To further manage this tradeoff, we train each booster for $\mathrm{B}_{\min} \leq B_{\text{es}} \leq B_{\max}$ rounds; B_{es} is the optimal number of rounds found through early stopping based on a single fixed i.i.d. validation noise realization. Here, $\mathrm{B}_{\min}$ is the minimum number of boosting rounds before early stopping is enforced; this is necessary to ensure a sufficient minimum number of noise realizations. This is a new custom implementation in our XGBoost fork[1]. $\mathrm{B}_{\max}$ is the maximum number of boosting rounds. To set η as low as possible while still yielding comparable total model capacity, we adaptively scale η down: $\eta = (\mathrm{B}_{\text{default}} \cdot \eta_{\text{default}})/\mathrm{B}_{\max}$, with $(\mathrm{B}_{\text{default}} = 100, \eta_{\text{default}} = 0.3)$ according to XGBoost defaults. Figure 3 provides an illustration of its usefulness: for low $\mathrm{B}_{\max}$ it can be beneficial to choose a higher learning rate, whereas a low learning rate paired with high $\mathrm{B}_{\max}$ yields best results.

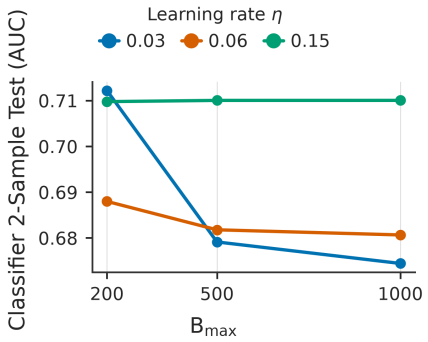

Figure 3: **Lowering η for high $\mathrm{B}_{\max}$ can be beneficial.** Average results for 6 small datasets. Full results in Figure 8.

Gaussian diffusion for numerical features Assuming a Gaussian latent variable such that $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$, the forward noising process is defined by

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{6}$$

where β_t defines the stepwise variance of a cosine variance-preserving schedule (Nichol & Dhariwal, 2021). With $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, the marginal forward process can be written as

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right), \tag{7}$$

or

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{8}$$

The reverse process is modelled by Gaussian transitions such that

$$\begin{aligned} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \tilde{\beta}_t \mathbf{I}), \\ \mu_\theta(\mathbf{x}_t, t) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_\theta(\mathbf{x}_t, t) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t, \end{aligned} \tag{9}$$

where $\tilde{\beta}_t = \dfrac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, and $\mathbf{x}_\theta(\mathbf{x}_t, t)$ is the prediction of the original data $\mathbf{x}_0$ at timestep t by a regression model parameterized by θ. Equivalently, we can reparameterize the model to predict the noise instead (Ho et al., 2020)

$$\epsilon_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_\theta(\mathbf{x}_t, t)), \tag{10}$$

or a velocity-based objective (Salimans & Ho, 2022), which tends to yield smoother loss trajectories (Gao et al., 2024), which we use:

$$v_\theta(\mathbf{x}_t, t) = \sqrt{\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t, t) - \sqrt{1-\bar{\alpha}_t}\mathbf{x}_\theta(\mathbf{x}_t, t). \tag{11}$$

We prefer to use a relatively low number of timesteps T, since we train a separate XGBoost model for each timestep. DDIMs are a generalization of DDPMs to non-Markovian diffusion processes, which yield deterministic generative processes capable of producing high quality samples much faster (Song et al., 2021). DDIMs have the same training objective as DDPMs, and are thus an easily implementable solution to our low T restriction. The deterministic DDIM reverse update can be defined as

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_\theta(\mathbf{x}_t, t) + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(\mathbf{x}_t, t). \tag{12}$$

Multinomial diffusion for categorical features Multinomial diffusion can be more suitable for modelling categorical data. For a categorical feature with K categories, let $x_t \in \{0,1\}^K$ denote its one-hot representation. We assume a uniform categorical latent variable $p(x_T) = \mathrm{Cat}(x_T; \frac{1}{K}\mathbf{1})$, such that the forward noising process adds uniform noise over the K categories (Hoogeboom et al., 2021):

$$q(x_t|x_{t-1}) = \mathrm{Cat}\left(x_t; \alpha_t x_{t-1} + \frac{1-\alpha_t}{K}\mathbf{1}\right). \tag{13}$$

The corresponding marginal distribution is

$$q(x_t|x_0) = \mathrm{Cat}\left(x_t; \bar{\alpha}_t x_0 + \frac{1-\bar{\alpha}_t}{K}\mathbf{1}\right). \tag{14}$$

Correspondingly, the reverse process is modelled by categorical transitions such that

$$\begin{aligned} p_\theta(x_{t-1}|x_t) &= \mathrm{Cat}\left(x_{t-1}; \frac{\boldsymbol{\pi}_\theta(\mathbf{x}_t, t)}{\sum_{k=1}^K \pi_{k,\theta}(\mathbf{x}_t, t)}\right), \\ \boldsymbol{\pi}_\theta(\mathbf{x}_t, t) &= \left[\alpha_t x_t + \frac{1-\alpha_t}{K}\mathbf{1}\right] \odot \left[\bar{\alpha}_{t-1}\mathbf{x}_\theta(\mathbf{x}_t, t) + \frac{1-\bar{\alpha}_{t-1}}{K}\mathbf{1}\right], \end{aligned} \tag{15}$$

where $\mathbf{x}_\theta(\mathbf{x}_t, t)$ is the predicted probability vector for the original category x_0 at timestep t by a classifier parameterized by θ. Equation (13) can equivalently be implemented as a two-step process: with probability α_t we retain the category at the previous timestep, and with probability $1-\alpha_t$ we resample uniformly from the K categories. We use this representation to allow us to retain categorical features as integer-encoded values instead of one-hot encoded vectors; this is useful since XGBoost can directly split on integer-encoded categories.

Combining Gaussian and multinomial diffusion We model numerical features through Gaussian diffusion and categorical features through multinomial diffusion. TabDDPM (Kotelnikov et al., 2023) is a previous neural diffusion model combining Gaussian and multinomial diffusion. In TabDDPM, numerical and categorical likelihoods contribute to a shared loss, while individual terms may lie on a different scale, potentially causing gradients to be dominated by high-variance or high-cardinality features. Some works go to great lengths to weight the different terms in the loss function (Mueller et al., 2025). Our method circumvents the need for loss-weighting as we train separate XGBoost models per feature: at each timestep t, we parameterize θ_t^j by an XGBoost regressor if feature j is numerical or by an XGBoost classifier if feature j is categorical, conditioning on all features of the current noised observation $\mathbf{x}_t$. During training, we jointly noise all features to timestep t according to their corresponding forward processes. Each feature-wise model then predicts its target from the complete mixed-type noised observation $\mathbf{x}_t$. At inference-time, numerical features are initialized from a Gaussian prior and categorical features from a uniform categorical prior. We subsequently apply the reverse transitions for each feature at every timestep, conditioning each feature-wise update on the same mixed-type state $\mathbf{x}_t$ until obtaining $\mathbf{x}_0$.

Preprocessing and hyperparameters. Numerical features are Gaussianized by a quantile transformation, categorical features are integer-encoded, and we use XGBoost's native functionality to split directly on categories. Following ForestDiffusion (Jolicoeur-Martineau et al., 2024) we use 50 timesteps. We train each XGBoost model using $\mathrm{B}_{\min} = 50$, which corresponds to the lower end of the dataset duplication factor in ForestDiffusion. We use 50 early stopping rounds, and adaptively set $\mathrm{B}_{\max}$ based on the dataset size, i.e., $\mathrm{B}_{\max} = 1000$ for $N <$ 5k, $\mathrm{B}_{\max} = 500$ for 5k $\leq N \leq$ 30k, and $\mathrm{B}_{\max} = 200$ for $N >$ 30k. We can interpret this adaptive training budget as somewhat similar to training deep generative models for a fixed number of training steps and batch size, yielding fewer full passes through the training data (epochs) on larger datasets. Our choice for $\mathrm{B}_{\max}$ here is based on our available hardware and experimental datasets, but should in practice be set as high as possible. Recall that learning rate η is set adaptively based on $\mathrm{B}_{\max}$. Other hyperparameters are kept as XGBoost's defaults, to further test the claim that GBDT-based diffusion models are relatively robust to hyperparameter choices (Section 1). In practice these can be tuned to further improve results.

4 Experiments

Baseline models. We compare XGB-DDPM against popular methods from literature: CTGAN and TVAE (Xu et al., 2019), ARF (Watson et al., 2023), TabDDPM (Kotelnikov et al., 2023), TabSyn (Zhang et al., 2024), TabDiff (Shi et al., 2025), CDTD (Mueller et al., 2025), ForestDiffusion, and ForestFlow (Jolicoeur-Martineau et al., 2024). Here, we align architectures and hyperparameters of deep generative models following previous work to ensure a similar training budget (Mueller et al., 2025; 2026). We provide an ablation w.r.t. the default hyperparameters suggested in the original works (Appendix B.1). Implementation details are provided in Appendix A.2, details on hardware are provided in Appendix A.4.

Evaluation. We use a variety of metrics which evaluate different quality dimensions of the generated data. We measure univariate fidelity through average Wasserstein Distance (WD) between real/generated numerical features and average Jensen-Shannon Distance (JSD) between real/generated categorical features. We measure bivariate fidelity through similarity in weighted normalized Mutual Information (MI) (Scassola et al., 2026). We measure multivariate fidelity through a Classifier 2-Sample Test (C2ST) (Friedman, 2004; Lopez-Paz & Oquab, 2017). We distinguish realism and coverage of the generative distribution through α-Precision and β-Recall (Alaa et al., 2022). We measure real-world machine learning utility through Machine Learning Efficacy (Shi et al., 2025). We measure the degree of overfitting through DCR Share (Platzer & Reutterer, 2021), which indicates the proportion of generated data closer to the training set than to the held-out test set. Finally, we measure membership disclosure risk through the Lift@10% of a classifier-based Membership Inference Attack (MIA). Here, Lift@10% indicates the membership enrichment in the top 10% predictions, and thereby measures the increased risk for the most at-risk observations. However, we do note that privacy is not our main concern, and rigorous privacy evaluation would include additional context-specific metrics and potential inclusion of privacy-preserving mechanisms (e.g., differential-privacy). Still, an MIA can be useful as it indicates local overfitting (van Breugel et al.,

2023) and thereby is an indication of generative quality. Further details on the evaluation metrics can be found in Appendix A.3.

Datasets. We include 10 mixed-type tabular datasets from TabArena (Erickson et al., 2026): `diabetes`, `credit-g`, `fiat500`, `wine`, `jm1`, `shipping`, `default`, `bank`, `diamonds`, `give-credit` (Appendix A.1). We randomly split the data into 80:20 train/test sets. We use 3 training seeds which define the data split, model initialization, and randomness in preprocessing. For each trained model we use 10 sampling seeds to query datasets at the size of the training set. We cap real and generated datasets at 50k rows during evaluation to limit runtime.

4.1 Results: Comparison to State-of-the-Art Tabular Generative Models

Table 1 aggregates metric scores over the datasets, generators, and seeds. We convert the metrics to the error rates w.r.t. the best potential scores for the respective metrics (Appendix A.3). This ensures a more intuitive comparison: it shows the improvability for each metric, and the direction is consistent (lower=better). Table 2 provides the corresponding average training and sampling times. Raw metric scores and training/sampling times for each dataset can be found in Appendix B.7.

Table 1: Error rates of metric scores. Shows $\mu \pm \sigma$ across 10 datasets. Per-dataset results are averaged over 3 training seeds and 10 sampling seeds. Best results in **blue bold**, second best underlined. Metrics with the (test) label have been computed w.r.t. the test set.

	C2ST	C2ST (test)	WD (num.)	JSD (cat.)	MI	α-Precision	β-Recall	MLE	DCR Share	MIA Lift
ARF	$37.1_{\pm 12.2}$	$32.4_{\pm 13.2}$	$1.1_{\pm 1.2}$	**$0.8_{\pm 0.7}$**	$5.3_{\pm 3.2}$	**$3.3_{\pm 2.5}$**	$15.2_{\pm 12.3}$	$6.0_{\pm 6.6}$	$3.1_{\pm 1.7}$	$4.8_{\pm 2.4}$
CTGAN	$46.4_{\pm 4.0}$	$43.8_{\pm 6.5}$	$2.6_{\pm 2.2}$	$18.3_{\pm 9.4}$	$9.0_{\pm 9.6}$	$20.1_{\pm 9.6}$	$39.7_{\pm 22.8}$	$14.4_{\pm 12.6}$	$1.6_{\pm 1.4}$	**$2.8_{\pm 2.7}$**
TVAE	$43.0_{\pm 12.0}$	$40.8_{\pm 13.0}$	$2.8_{\pm 2.9}$	$14.3_{\pm 9.7}$	$9.1_{\pm 10.4}$	$18.4_{\pm 15.4}$	$52.5_{\pm 32.2}$	$16.3_{\pm 11.7}$	$5.7_{\pm 6.1}$	$5.6_{\pm 6.5}$
ForestDiffusion	$41.1_{\pm 15.3}$	$38.9_{\pm 16.4}$	$1.2_{\pm 0.6}$	$2.8_{\pm 3.6}$	$12.4_{\pm 11.6}$	$10.5_{\pm 10.5}$	$47.9_{\pm 34.7}$	$10.9_{\pm 11.1}$	$2.4_{\pm 2.5}$	$3.0_{\pm 2.6}$
ForestFlow	$38.7_{\pm 17.7}$	$37.2_{\pm 16.7}$	$0.7_{\pm 0.7}$	$2.8_{\pm 3.6}$	$3.2_{\pm 2.7}$	$6.3_{\pm 3.2}$	$17.9_{\pm 19.8}$	$8.1_{\pm 7.7}$	$2.7_{\pm 3.0}$	$3.9_{\pm 3.6}$
TabDDPM	$18.5_{\pm 16.5}$	$13.8_{\pm 9.9}$	$1.0_{\pm 1.2}$	$1.9_{\pm 1.9}$	$2.8_{\pm 2.5}$	$3.5_{\pm 2.9}$	$6.2_{\pm 4.7}$	$4.8_{\pm 5.4}$	$4.2_{\pm 5.1}$	$10.8_{\pm 13.5}$
TabSyn	$32.2_{\pm 14.6}$	$25.1_{\pm 13.5}$	$2.2_{\pm 3.5}$	$2.3_{\pm 2.5}$	$5.2_{\pm 6.3}$	$12.9_{\pm 15.9}$	$21.4_{\pm 24.3}$	$5.7_{\pm 5.6}$	$1.7_{\pm 2.1}$	$4.0_{\pm 4.5}$
TabDiff	$35.0_{\pm 13.0}$	$29.8_{\pm 13.4}$	$2.4_{\pm 2.4}$	$1.3_{\pm 1.0}$	$8.1_{\pm 11.5}$	$15.3_{\pm 17.1}$	$30.1_{\pm 28.1}$	$8.7_{\pm 10.4}$	**$1.4_{\pm 1.2}$**	$3.6_{\pm 4.4}$
CDTD	**$17.8_{\pm 14.4}$**	$13.4_{\pm 7.1}$	**$0.6_{\pm 0.5}$**	$1.1_{\pm 0.7}$	**$2.2_{\pm 2.5}$**	$3.4_{\pm 1.8}$	**$2.9_{\pm 3.5}$**	**$3.4_{\pm 3.3}$**	$6.7_{\pm 5.5}$	$12.6_{\pm 11.2}$
XGB-DDPM	$18.2_{\pm 14.4}$	**$12.3_{\pm 7.3}$**	**$0.6_{\pm 0.4}$**	$1.4_{\pm 1.1}$	$4.4_{\pm 6.2}$	$5.2_{\pm 3.2}$	$7.6_{\pm 6.0}$	$4.1_{\pm 4.5}$	$2.2_{\pm 2.8}$	$5.0_{\pm 2.7}$

State-of-the-art fidelity of tabular diffusion model on the CPU. Table 1 indicates that XGB-DDPM performs on par in terms of multivariate fidelity (C2ST) with state-of-the-art diffusion models, at reasonable training and sampling times (Table 2), while running entirely on CPUs. The multivariate test set fidelity (C2ST (test)) even scores best out of all the baselines, indicating a best-generalizing generative distribution. XGB-DDPM similarly performs strongly on univariate and bivariate fidelity measures (WD, JSD, MI), realism and coverage (α-Precision, β-Recall), and real-world machine learning utility (MLE).

Table 2: Training and sampling times (Table 1).

	Training Time (min.)	Sampling Time (sec.)
ARF	**$0.3_{\pm 0.4}$**	$0.3_{\pm 0.3}$
CTGAN	$15.1_{\pm 1.4}$	**$0.1_{\pm 0.1}$**
TVAE	$8.1_{\pm 3.1}$	**$0.1_{\pm 0.1}$**
ForestDiffusion	$17.6_{\pm 19.1}$	$1.5_{\pm 1.6}$
ForestFlow	$12.0_{\pm 12.8}$	$1.1_{\pm 1.2}$
TabDDPM	$3.3_{\pm 1.2}$	$17.9_{\pm 19.8}$
TabSyn	$4.3_{\pm 1.4}$	$0.7_{\pm 0.6}$
TabDiff	$4.5_{\pm 0.9}$	$1.4_{\pm 1.3}$
CDTD	$3.7_{\pm 1.1}$	$1.7_{\pm 1.6}$
XGB-DDPM	$4.9_{\pm 3.4}$	$11.1_{\pm 14.1}$

The frontier between fidelity, overfitting, and membership disclosure. Crucially, high fidelity of the generative distribution should not come at the expense of an increased risk of overfitting, either globally or locally. Especially local overfitting can be a source of privacy risk through membership disclosure (van Breugel et al., 2023). XGB-DDPM displays lower overfitting risk (DCR Share) and membership disclosure risk (MIA Lift) than other high-fidelity baselines (e.g., TabDDPM and CDTD). This further suggests that XGB-DDPM learns accurate and generalizable generative distributions, as already hinted at by scoring best in terms of multivariate fidelity with respect to the test set (C2ST (test)). Figure 4 provides scatterplots for further intuition.

Scaling to million-row datasets: results for the poker-hand dataset. Next, we investigate whether XGB-DDPM can be effectively trained within a reasonable time on datasets with a very large number of observations. Unfortunately, the largest dataset from TabArena has "just" 150k observations (`give-credit`). We therefore source the dataset with most observations from the TabZilla benchmark (McElfresh et al., 2023) instead: `poker-hand`, which has 1.025M observations and 11 features. Table 3 provides the results for the `poker-hand` dataset. XGB-DDPM completed training in $\approx$ 33 minutes on average, and achieves highest fidelity out of all the baselines. This further exemplifies that XGB-DDPM can be effectively trained on datasets with a large number of observations. In contrast, ForestDiffusion and ForestFlow could not be trained due to out-of-memory errors on our hardware (128GB RAM).

Table 3: Error rates of metric scores for the **poker-hand** dataset. Shows average $\pm$ standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **blue bolded** and second best scores are underlined. ForestDiffusion and ForestFlow are out-of-memory during training.

	C2ST	C2ST (test)	WD (num.)	JSD (cat.)	MI	α-Precision	β-Recall	MLE	DCR Share	MIA Lift
ARF	$15.6_{\pm 0.4}$	$15.7_{\pm 0.4}$	$1.3_{\pm 0.0}$	**$0.8_{\pm 0.2}$**	**$0.0_{\pm 0.0}$**	$2.4_{\pm 0.2}$	$3.4_{\pm 0.1}$	$11.2_{\pm 4.9}$	$1.2_{\pm 0.2}$	$1.2_{\pm 0.8}$
CTGAN	$43.1_{\pm 0.5}$	$43.2_{\pm 0.5}$	$3.8_{\pm 0.4}$	$58.0_{\pm 0.2}$	$1.5_{\pm 0.1}$	$21.9_{\pm 3.2}$	$25.3_{\pm 0.8}$	$48.0_{\pm 3.5}$	$0.3_{\pm 0.3}$	$0.5_{\pm 0.7}$
TVAE	$50.0_{\pm 0.0}$	$50.0_{\pm 0.0}$	$30.5_{\pm 7.4}$	$43.7_{\pm 17.2}$	$7.7_{\pm 10.9}$	$42.5_{\pm 26.0}$	$98.8_{\pm 1.7}$	$58.0_{\pm 4.5}$	$14.5_{\pm 18.9}$	$0.4_{\pm 0.5}$
ForestDiffusion	-	-	-	-	-	-	-	-	-	-
ForestFlow	-	-	-	-	-	-	-	-	-	-
TabDDPM	$7.9_{\pm 0.4}$	$8.0_{\pm 0.3}$	**$0.3_{\pm 0.1}$**	**$0.8_{\pm 0.1}$**	**$0.0_{\pm 0.0}$**	**$1.3_{\pm 0.4}$**	$2.7_{\pm 0.1}$	$11.9_{\pm 4.8}$	**$0.1_{\pm 0.2}$**	$0.4_{\pm 0.6}$
TabSyn	$9.6_{\pm 0.7}$	$9.7_{\pm 0.4}$	$0.9_{\pm 0.2}$	$1.7_{\pm 0.2}$	**$0.0_{\pm 0.0}$**	$1.4_{\pm 0.7}$	$3.0_{\pm 0.1}$	$27.3_{\pm 4.8}$	**$0.1_{\pm 0.2}$**	**$0.3_{\pm 0.5}$**
TabDiff	$48.6_{\pm 0.4}$	$48.6_{\pm 0.4}$	$15.6_{\pm 3.4}$	$43.4_{\pm 9.3}$	**$0.0_{\pm 0.0}$**	$89.7_{\pm 0.6}$	$64.3_{\pm 2.7}$	$44.7_{\pm 4.8}$	$0.2_{\pm 0.3}$	$0.4_{\pm 0.5}$
CDTD	$39.9_{\pm 0.1}$	$39.9_{\pm 0.2}$	$6.2_{\pm 0.1}$	$2.9_{\pm 1.0}$	**$0.0_{\pm 0.0}$**	$57.6_{\pm 0.6}$	$34.6_{\pm 0.3}$	$32.2_{\pm 4.5}$	$0.2_{\pm 0.3}$	$0.5_{\pm 0.6}$
XGB-DDPM	**$2.6_{\pm 0.9}$**	**$2.9_{\pm 0.6}$**	**$0.3_{\pm 0.0}$**	**$0.8_{\pm 0.1}$**	**$0.0_{\pm 0.0}$**	$2.3_{\pm 0.4}$	**$2.5_{\pm 0.1}$**	**$10.6_{\pm 4.8}$**	**$0.1_{\pm 0.1}$**	$0.4_{\pm 0.5}$

4.2 Ablation Studies

Table 4 shows the incremental effect of the different ingredients of XGB-DDPM. Here, we start from a "Base" version: XGB-DDPM with fully numerical diffusion (categorical features are one-hot encoded as in ForestDiffusion), a single noise realization per observation/timestep, and a stochastic (DDPM) sampler, ϵ-objective, and linear noise schedule (Ho et al., 2020). We subsequently add multinomial diffusion for categorical features (+Cat. diffusion), refresh the noise realization across boosting rounds (+Noise refresh), and include our deterministic sampler, velocity-objective, and cosine noise schedule (+DDIM, v-pred., cos. sched.). As expected, adding multinomial diffusion enhances fidelity of categorical features (JSD (cat.)), and refreshing the noise realizations majorly enhances multivariate fidelity (C2ST AUC). Finally, configuring the diffusion objective, sampler, and noise schedule provides another fidelity boost.

Table 4: Average Δ in metric error rates compared to the "Base" version of XGB-DDPM for each (seed-averaged) dataset-wise comparison, including 95% CI. Improvements are additive such that each column also includes the improvements to its left. Insignificant differences (0 contained in CI) in black while significant improvements and degradations are colored. The grey column indicates our full method.

% $\downarrow$	Base	+Cat. diffusion	+Noise refresh	+DDIM, v-pred., cos. sched.
C2ST	27.3 [21.0, 33.6]	-2.5 [-5.7, $+0.7$]	-8.0 [-13.2, -2.8]	-9.0 [-14.5, -3.6]
C2ST (test)	18.3 [14.6, 21.9]	-2.6 [-5.5, $+0.4$]	-4.8 [-7.6, -2.0]	-6.0 [-8.8, -3.2]
WD (num.)	1.0 [0.6, 1.4]	$+0.2$ [-0.2, $+0.5$]	-0.2 [-0.4, $+0.0$]	-0.4 [-0.6, -0.2]
JSD (cat.)	3.7 [2.1, 5.3]	-2.3 [-3.5, -1.1]	-2.3 [-3.4, -1.2]	-2.3 [-3.5, -1.2]
MI	6.4 [1.9, 10.9]	-0.4 [-0.9, $+0.2$]	-2.2 [-3.3, -1.2]	-1.9 [-3.1, -0.8]
α-Precision	8.4 [6.6, 10.3]	-0.4 [-3.0, $+2.2$]	-0.4 [-2.9, $+2.1$]	-3.2 [-6.0, -0.4]
β-Recall	12.4 [7.9, 16.9]	-1.6 [-3.9, $+0.7$]	-5.9 [-9.3, -2.5]	-4.7 [-8.4, -1.1]
MLE	9.9 [3.6, 16.1]	-5.1 [-10.4, $+0.3$]	-6.0 [-11.3, -0.8]	-5.8 [-10.9, -0.6]
DCR Share	1.1 [0.4, 1.7]	$+0.3$ [-0.0, $+0.6$]	$+1.3$ [$+0.0$, $+2.6$]	$+1.2$ [-0.0, $+2.3$]
MIA Lift	3.7 [2.1, 5.2]	$+0.7$ [-0.2, $+1.6$]	$+1.9$ [$+0.2$, $+3.6$]	$+1.3$ [-0.0, $+2.6$]

4.2.1 DIFFUSION OBJECTIVES, SAMPLERS, AND NOISE SCHEDULE

XGB-DDPM can be configured to predict the noise (ϵ) or the velocity (v), use deterministic (DDIM) or non-deterministic (DDPM) sampling, and use a linear or cosine noise schedule. Figure 5 provides a three-way interaction plot of the different configurations. We notice poor performance of the noise-prediction objective when using a deterministic sampler and cosine noise schedule. This likely stems from error amplification at the especially low signal-to-noise ratios for the cosine schedule near the endpoint, which are then deterministically propagated through the sampling trajectory. We find a velocity-based parameterization with deterministic sampler and cosine noise schedule yields strong and balanced results.

4.2.2 TRAINING MODELS PER TIMESTEP AND TARGET LABEL

Following ForestDiffusion (Jolicoeur-Martineau et al., 2024) we train separate XGBoost models for each timestep and categorical target label. Essentially, this forces conditioning on the timestep and label and therefore makes the modelling task easier. Additionally, it ensures that each XGBoost model gets trained on less data, which is especially beneficial in ForestDiffusion to reduce memory load. However, since we do not duplicate the dataset and memory load is much reduced, we can evaluate whether it is feasible to add the timestep as an additional feature to the XGBoost models, and perform unconditional generation. Figure 6 provides an ablation of training separate XGBoost models for each timestep and target label versus adding them as input features. Overall, training separate XGBoost models per timestep enhances fidelity and majorly decreases training time, especially for larger datasets. The gains are less clear for also training separate models per target label.

4.2.3 MEMORY USAGE, TRAINING TIME, AND SAMPLING TIME

We provide a detailed evaluation of resource usage scaling with data size in Figure 7. For each plot we vary either the number of observations, features, or target labels from their defaults $n = 1000, d = 10, c = 2$. Features are generated randomly from $U(0, 1)$, and target labels are randomly-assigned integers. As expected, XGB-DDPM is far more efficient in terms of peak memory burden. It scales especially well with increasing observations compared to ForestDiffusion: for increasing n the data takes up a larger proportion of the total memory pressure, in which case the dataset duplication of ForestDiffusion becomes especially troublesome. For increasing features and target labels the dataset duplication does not play a role, and the difference between XGB-DDPM and ForestDiffusion remains relatively constant. In terms of training time, XGB-DDPM scales especially well with increasing n due to not relying on dataset duplication. For very large number of features and target classes, however, ForestDiffusion can be more efficient. More features and target classes mean more histograms have to be updated at each boosting round, which increases training time. Lastly, ForestDiffusion is more efficient at sampling than XGB-DDPM. This mainly stems from the fact that ForestDiffusion can use a single multi-output XGBoost regressor at each timestep and target class, whereas XGB-DDPM has to loop (in parallel) over each feature using a separate XGBoost estimator.

5 CONCLUSION AND DISCUSSION

Refreshing noise realizations across boosting rounds in GBDTs is a cost-effective strategy to optimize stochastic denoising objectives. We implemented this method in an XGBoost fork, and showcased its effectiveness in XGB-DDPM: a DDPM-style diffusion model which can be efficiently trained on large datasets using CPUs. We further showed that XGB-DDPM performs on par with neural tabular diffusion models in terms of generative quality, and is therefore a useful alternative for practitioners who do not have easy access to GPUs.

XGB-DDPM trains separate XGBoost models per feature. This makes it especially efficient for "long and narrow" datasets, as often found in, e.g., transactional and event tables. It is less efficient for datasets with many features. Furthermore, this work is especially geared towards solving the memory and runtime costs of GBDT-based diffusion models. The diffusion specification can be improved in several ways to yield even better results, e.g., by using more sophisticated diffusion kernels and noise schedules.

AI USE STATEMENT

We used generative AI tools to implement methods, design or provide feedback on research methodology or experiments, and assist in the writing of proofs. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results. Additionally, we used generative AI tools to create or edit software code and for brainstorming. We have reviewed all AI-assisted work. That is, AI-generated code was manually reviewed and verified by 2 of the authors, and any feedback on methods, proofs, or brainstorming was discussed with (at least) 2 of the authors as well. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ahmed Alaa, Boris Van Breugel, Evgeny S Saveliev, and Mihaela Van Der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International conference on machine learning*, pp. 290–306. PMLR, 2022.

Barry Becker and Ronny Kohavi. Adult, 1996.

Nicolas Beltran Velez, Alessandro A Grande, Achille Nazaret, Alp Kucukelbir, and David Blei. Treeffuser: probabilistic prediction via conditional diffusions with gradient-boosted trees. *Advances in Neural Information Processing Systems*, 37:118296–118325, 2024.

Bernd Bischl, Giuseppe Casalicchio, Taniya Das, Matthias Feurer, Sebastian Fischer, Pieter Gijsbers, Subhaditya Mukherjee, Andreas C Müller, László Németh, Luis Oala, Lennart Purucker, Sahithya Ravi, Jan N van Rijn, Prabhant Singh, Joaquin Vanschoren, Jos van der Velde, and Marcel Wever. Openml: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7):101317, 2025. doi: 10.1016/j.patter.2025.101317. URL https://www.cell.com/patterns/fulltext/S2666-3899(25)00165-5.

João Bravo. NRGBoost: Energy-based generative boosted trees. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=wQHyjIZ1SH.

Jesse C Cresswell and Taewoo Kim. Scaling up diffusion and flow-based xgboost models. *arXiv preprint arXiv:2408.16046*, 2024.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *Advances in Neural Information Processing Systems*, 38, 2026.

Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

Jerome Friedman. On multivariate goodness-of-fit and two-sample testing. Technical report, SLAC National Accelerator Laboratory (SLAC), Menlo Park, CA (United States), 2004.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4): 367–378, 2002.

Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P Murphy, and Tim Salimans. Diffusion meets flow matching: Two sides of the same coin. 2024. https://diffusionflow.github.io, 2024. Accessed: 2026-01-07.

John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pp. 857–871, 1971.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520, 2022.

Andrés Guzmán-Cordero, Floor Eijkelboom, and Jan-Willem van de Meent. Exponential family variational flow matching for tabular data generation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=kjtvCSkSsy.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in neural information processing systems*, 34:12454–12465, 2021.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 1288–1296. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/jolicoeur-martineau24a.html.

Jayoung Kim, Chaejeong Lee, and Noseong Park. STasy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1mNssCWt_v.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *International conference on machine learning*, pp. 17564–17579. PMLR, 2023.

Anton D Lautrup, Tobias Hyrup, Arthur Zimek, and Peter Schneider-Kamp. Syntheval: a framework for detailed utility and privacy evaluation of tabular synthetic data: Ad lautrup et al. *Data Mining and Knowledge Discovery*, 39(1):6, 2025.

Vili Lehdonvirta, Bóxī Wú, and Zoe Hawkins. Compute north vs. compute south: the uneven possibilities of compute-based ai governance around the globe. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 7, pp. 828–838, 2024.

David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJkXfE5xx.

Calvin McCarter. Unmasking trees for tabular data. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. URL https://openreview.net/forum?id=LFCSTy6MYe.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in neural information processing systems*, 36:76336–76369, 2023.

Markus Mueller, Kathrin Gruber, and Dennis Fok. Continuous diffusion for mixed-type tabular data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=QPtoBPn4lZ.

Markus Mueller, Kathrin Gruber, and Dennis Fok. Cascaded flow matching for heterogeneous tabular data with mixed-type features. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=l2ywV9sV0L.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Michael Platzer and Thomas Reutterer. Holdout-based empirical assessment of mixed-type synthetic data. *Frontiers in big Data*, 4:679939, 2021.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Davide Scassola, Dylan Ponsford, Adrián Javaloy, Sebastiano Saccani, Luca Bortolussi, Henry Gouk, and Antonio Vergari. A sobering look at tabular data generation via probabilistic circuits. In Emilija Perković and Daniel Malinsky (eds.), *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 6026–6063. PMLR, 17–21 Aug 2026. URL https://proceedings.mlr.press/v337/scassola26a.html.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. Tabdiff: a mixed-type diffusion model for tabular data generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=swvURjrt8z.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Boris van Breugel, Hao Sun, Zhaozhi Qian, and Mihaela van der Schaar. Membership inference attacks against synthetic data through overfitting detection. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 3493–3514. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/breugel23a.html.

Joshua Ward, Yuxuan Yang, Chi-Hua Wang, and Guang Cheng. Ensembling membership inference attacks against tabular generative models. In *Proceedings of the 18th ACM Workshop on Artificial Intelligence and Security*, pp. 182–193, 2025.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 5357–5375. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/watson23a.html.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

Zexi Yao, Nataša Krčo, Georgi Ganev, and Yves-Alexandre de Montjoye. The dcr delusion: Measuring the privacy risk of synthetic data. In *European Symposium on Research in Computer Security*, pp. 469–487. Springer, 2025.

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

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan: Effective table data synthesizing. In *Asian conference on machine learning*, pp. 97–112. PMLR, 2021.

A EXPERIMENT DETAILS

A.1 DATASETS AND PREPROCESSING

Table 5 summarizes the experimental datasets. We source all datasets from the TabArena (Erickson et al., 2026) and TabZilla (McElfresh et al., 2023) benchmark, which ensures that all datasets are tabular i.i.d., have a supervised prediction task (which is useful for the MLE metric), and are publicly available via OpenML (Bischl et al., 2025).

Table 5: Overview of 10 benchmark datasets (Table 1) plus `poker-hand` (Table 3). Total cardinality counts the cumulative number of categories across all categorical features. Absent target classes (-) indicates regression task.

Dataset	Short name	OpenML ID	$N \cdot 10^3$	d	%categorical	Total cardinality	Target classes
diabetes	`diabetes`	46921	0.8	9	11.1	2	2
credit-g	`credit-g`	46918	1.0	21	66.7	56	2
Another-Dataset-on-used-Fiat-500	`fiat500`	46907	1.5	8	12.5	3	-
wine_quality	`wine`	46964	6.5	13	7.7	2	-
jm1	`jm1`	46979	10.9	22	4.5	2	2
E-CommerceShippingData	`shipping`	46924	11.0	11	45.5	15	2
credit-card-clients-default	`default`	46919	30.0	24	16.7	15	2
bank-marketing	`bank`	46910	45.2	14	64.3	34	2
diamonds	`diamonds`	46923	53.9	10	30.0	20	-
GiveMeSomeCredit	`give-credit`	46929	150.0	11	9.1	2	2
poker-hand	`poker`	1567	1025.0	11	9.1	10	10

Regarding missing data, we encode missing categorical data as separate categories. For numerical missing data, we impute missing values using MissForest (Stekhoven & Bühlmann, 2012); only `give-credit` has missing numerical data ($\approx 20\%$ of the rows). Lastly, we round numerical features in generated data to the precision of the original data.

A.2 BASELINE MODELS

Below we provide further details on baseline model implementations and hyperparameter choices. Here, we mostly wrap the official implementations of the original works.

ARF. Adversarial generative model using Random Forests in alternating rounds of generation and discrimination for density estimation and data generation (Watson et al., 2023). The implementation is based on the `arfpy` Python package (version 0.1.1), but uses an equivalent NumPy-vectorized sampling path. We use the default hyperparameters suggested in the original paper.

CTGAN. Conditional GAN using mode-specific normalization (Xu et al., 2019). The implementation is based on the `ctgan` Python package (version 0.12.0). We align the hyperparameters according to the hyperparameter alignment strategy. The generator and discriminator are enlarged in equal proportion as to not alter relative capacity. We adjust the batch size downward such that it is divisible by 10, which is necessary due to using packing. For other hyperparameters, we rely on the defaults from the original paper.

TVAE. VAE-based counterpart of CTGAN from the same paper, using the same encoding strategy (Xu et al., 2019). The implementation is based on the `ctgan` Python package (version 0.12.0). We align the hyperparameters according to the hyperparameter alignment strategy. The encoder and decoder are enlarged in equal proportion as to not alter relative capacity. For other hyperparameters, we rely on the defaults from the original paper.

TabDDPM. Denoising diffusion probabilistic model combining Gaussian and multinomial diffusion (Kotelnikov et al., 2023). The implementation is based on the implementation in the `synthcity` Python package (version 0.2.12). Since TabDDPM does not suggest any default hyperparameters, we follow the hyperparameter alignment strategy, and rely on the defaults from `synthcity` for any other hyperparameters.

TabSyn. Latent diffusion model, training a diffusion model in the latent space of a transformer-based VAE (Zhang et al., 2024). The implementation is based on the original paper's implementation. In the hyperparameter alignment, we align the diffusion model network but keep the VAE as in the original implementation. We train the VAE and diffusion model for 30k total steps in a similar proportion as in the original paper. Otherwise, we rely on the defaults from the original paper.

TabDiff. Continuous diffusion model for mixed-type tabular data combining score-matching and masked diffusion, and using learnable noise schedules per feature (Shi et al., 2025). The implementation is based on the original paper's implementation. We align the hyperparameters according to the hyperparameter alignment strategy; for other hyperparameters, we rely on the defaults from the original paper.

CDTD. Continuous diffusion model for mixed-type tabular data combining score-matching and score interpolation, and using learnable noise schedules per feature-type (Mueller et al., 2025). The implementation is based on the original paper's implementation. We rely on the hyperparameters from the original paper, which corresponds exactly to our hyperparameter alignment procedure.

ForestDiffusion and ForestFlow. ForestDiffusion is a score-based generative model using XGBoost as denoiser. ForestFlow is its conditional flow-matching counterpart (Jolicoeur-Martineau et al., 2024). The implementation is based on the improved implementation by Cresswell & Kim (2024). This is a more memory- and runtime-efficient implementation of the original ForestDiffusion. We use a dataset duplication factor $K = 50$, and otherwise the default hyperparameters from the original ForestDiffusion. For the XGBoost models we use 20-round early stopping based on a single fixed i.i.d. validation noise realization, and otherwise the default parameters.

Aligning deep generative models. We evaluate the neural baseline models under a common computational regime to avoid disparate compute budgets influencing results. Specifically, we follow the alignment strategy from Mueller et al. (2025; 2026). This means that all models only use feedforward layers, that layers/nodes are aligned such that each model has ≈ 3.0M trainable parameters on the `adult` dataset (Becker & Kohavi, 1996), and that embedding/bottleneck dimensions are aligned to 256. Additionally, each model uses batch size 4096, and is trained for 30k total steps – or until the time cap of 1 hour is reached. This alignment strategy is used for CTGAN, TVAE, TabDDPM, TabSyn, TabDiff, and CDTD, unless mentioned otherwise in the description of the respective model implementation.

We also provide the results for the baseline models under their original default set-up as proposed in the original works (Appendix B.1). We find that this does not influence our conclusions.

A.3 Evaluation metrics

Classifier 2-Sample Test (C2ST). We evaluate multivariate fidelity through the ability of a machine learning classifier to distinguish real from generated data (Friedman, 2004; Lopez-Paz & Oquab, 2017). This is the same measure underlying the Detection score from the popular SDMetrics[2] library, which has been used in several relevant works (Zhang et al., 2024; Shi et al., 2025). To allow power against many alternative distributions, C2ST requires a powerful classifier (Friedman, 2004; Lopez-Paz & Oquab, 2017; Scassola et al., 2026). We use a GBDT classifier, specifically XGBoost. Across 3-fold cross-validation, we report the highest average AUC achieved on the validation sets over 500 boosting rounds. To compute C2ST on the held-out test set, generated data is subsampled to the test set size; the entire procedure is repeated 3 times and we average the scores.

We calculate the C2ST error rate as follows:

$$\text{C2ST}_{\text{err}\%} = 100 \cdot [\text{C2ST}_{\text{AUC}} - 0.5]_+,$$

This essentially caps the error rate: since the label orientation is fixed, any AUC < 0.5 should similarly be interpreted as no distinguishability. This notion is also used in SDMetrics' Detection score[2].

[2]https://docs.sdv.dev/sdmetrics

Wasserstein Distance (WD) and Jensen-Shannon Distance (JSD). We evaluate similarity in univariate distributions by measuring the average distance between real and generated features. For numerical features we use Wasserstein-1 distance on [0,1] scaled data, for categorical features we use Jensen-Shannon Distance. We compute the error rates as follows:

$$\text{WD}_{\text{err}\%} = 100 \cdot \text{WD}, \quad \text{JSD}_{\text{err}\%} = 100 \cdot \text{JSD}$$

Mutual Information (MI). We evaluate similarity in bivariate distributions of real and generated data through MI. Specifically, we use the weighted normalized MI from Scassola et al. (2026). This is a more informative bivariate fidelity measure than, e.g., difference in correlation matrix, since MI is sensitive to nonlinear associations and invariant to differences in the univariate distributions (Scassola et al., 2026). The MI error rate can be calculated as:

$$\text{MI}_{\text{err}\%} = 100 \cdot (1 - \text{MI})$$

α-Precision & β-Recall. We use the integrated α-Precision & β-Recall scores from Alaa et al. (2022) to determine whether the generated data is both realistic and covers the full range of the real distribution. In β-Recall, we use $k = 5$ to determine local coverage of the generated data by the real data, following the original work (Alaa et al., 2022). Distances are calculated using Gower's distance (Gower, 1971), to provide a balanced measure for mixed-type data. We calculate the precision and recall error rates as:

$$\alpha\text{-Precision}_{\text{err}\%} = 100 \cdot (1 - \alpha\text{-Precision}), \quad \beta\text{-Recall}_{\text{err}\%} = 100 \cdot (1 - \beta\text{-Recall})$$

Machine Learning Efficacy (MLE). We evaluate the usefulness of generated data to train supervised machine learning models, which are then evaluated on the held-out real data. This measure is sometimes also referred to as the Train Synthetic Test Real (TSTR) approach (Esteban et al., 2017). Similar to the C2ST, we use XGBoost models with 500 boosting rounds. We use 30-round early stopping based on an internal 15% validation set. We report R^2 for regression datasets and AUC for classification datasets. We compare the scores on the generated data with scores achieved from training on the real training data. We calculate a unified MLE error rate in the main results (Table 1) as follows:

$$\text{MLE}_{\text{err}\%} = 100 \cdot (w \cdot |M_G^* - M_R^*|),$$

$$M^* = \begin{cases} 0.5 + [M - 0.5]_+ \\ [M]_+ \end{cases}, \quad w = \begin{cases} 2 & \text{if } M = \text{AUC} \\ 1 & \text{if } M = R^2 \end{cases}$$

which again caps the error rate such that any degenerate score (AUC < 0.5 or $R^2 < 0$) is considered similarly, and ensures that errors in classification and regression datasets lie on a similar [0,1] scale.

Distance to Closest Record Share (DCR Share). We evaluate the degree to which the generated data overfits the training data through DCR Share (Platzer & Reutterer, 2021). Note that it should not be interpreted as a privacy measure (Yao et al., 2025). DCR Share corresponds to the proportion of generated data which is closer to the training data – in our case, in terms of Gower distance – than the held-out real test set. Generated data and training data are subsampled to N_{test} rows such that DCR Share $= 0.5$ indicates no overfitting; generated data is equally often closer to the training data and the held-out test set. We repeat this procedure $\lceil \frac{N_{\text{train}}}{N_{\text{test}}} \rceil$ times and average the results. We calculate the DCR Share error rate as:

$$\text{DCR Share}_{\text{err}\%} = 100 \cdot [\text{DCR Share} - 0.5]_+$$

since any DCR Share < 0.5 similarly indicates no sign of overfitting. This notion is also used in SDMetrics' DCROverfittingProtection[2].

Membership Inference Attack (MIA). We measure membership disclosure risk through a classifier-based MIA, using a set-up similar to SynthEval (Lautrup et al., 2025). Note that this is also the same attack set-up as DOMIAS (van Breugel et al., 2023), while using a classifier to estimate the density ratios instead of multivariate density estimators. We split the test set ($\mathcal{D}_{\text{test}}$) into reference data assumed to be available to the attacker ($\mathcal{D}_{\text{test}}^{\text{fit}}$) and non-members to the training data for evaluation ($\mathcal{D}_{\text{test}}^{\text{eval}}$) in 1:1 proportion. We then train a binary machine learning classifier to distinguish between the real reference data ($\mathcal{D}_{\text{test}}^{\text{fit}}$) and an equally-sized subsample of the generated data ($\mathcal{D}_{\text{gen}}^{(b)}$). Here, the goal is to teach the classifier unusual patterns from the training data which are mimicked in the generated data. The classifier is then evaluated on the non-members ($\mathcal{D}_{\text{test}}^{\text{eval}}$) and an equally-sized subsample of the training data ($\mathcal{D}_{\text{train}}^{(b)}$); for the correctly classified training samples, membership has been disclosed. We repeat the entire procedure 5 times to account for sampling variability and average the results.

We use a rank-averaged ensemble of a Random Forest and XGBoost classifier as an attack classifier. Ensembling MIAs has been found to provide more accurate and stable MIAs (Ward et al., 2025). As main performance measure we report the Lift@10%, which indicates membership enrichment in the top 10% predictions:

$$\text{MIA Lift@10\%} = \frac{\frac{1}{|\mathcal{T}_{0.1}|}\sum_{i\in\mathcal{T}_{0.1}} y_i}{\frac{1}{n}\sum_{i=1}^{n} y_i},$$

where $\mathcal{T}_{0.1}$ denotes the set containing the 10% observations with highest predicted membership scores. Intuitively, Lift@10% $= 1.5$ means the top 10% predictions contain members at $1.5\times$ the dataset-wide rate. This measure indicates the additional proportional membership disclosure risk over random guessing, but only for the most at-risk observations. This can be useful since membership disclosure typically only occurs for a small group of risky observations. We calculate the metric error rates as follows:

$$\text{MIA Lift}_{\text{err\%}} = 100 \cdot [\text{MIA Lift} - 1]_+$$

since any Lift < 1.0 similarly indicates no sign of additional membership disclosure risk. We also report the AUC of the MIA in the detailed results for completeness (Table 20).

A.4 Hardware

Neural network-based baseline models are trained using a single MIG-instance of an RTX6000 (24GB RAM) and 8 CPU cores (32GB RAM): CTGAN, TVAE, TabDDPM, TabSyn, TabDiff, CDTD. Non-neural models are trained using 32 CPU cores (128GB RAM): ARF, ForestDiffusion, ForestFlow, and XGB-DDPM.

B Additional results

B.1 Results under default baseline hyperparameters

We provide the main experimental results from Table 1 under the default baseline hyperparameters, instead of using the alignment strategy for the deep generative baseline models (Appendix A.2). Note that this only considers CTGAN, TVAE, TabSyn, and TabDiff: others already used default parameters (CDTD) or have no defaults available in the original work (TabDDPM). Results are given in Table 7.

Comparing Table 7 to Table 1, generative quality improves for TabSyn and TabDiff when using the original default hyperparameters. This does come at the cost of much larger training times (Table 6); also, both are still outperformed by TabDDPM, CDTD, and XGB-DDPM. We therefore still conclude

Table 6: Training and sampling times (Table 7).

	Training Time (min.)	Sampling Time (sec.)
ARF	**0.3**$_{\pm 0.4}$	0.3$_{\pm 0.3}$
CTGAN (org.)	3.3$_{\pm 4.5}$	**0.1**$_{\pm 0.1}$
TVAE (org.)	1.7$_{\pm 2.2}$	**0.1**$_{\pm 0.1}$
ForestDiffusion	17.6$_{\pm 19.1}$	1.5$_{\pm 1.6}$
ForestFlow	12.0$_{\pm 12.8}$	1.1$_{\pm 1.2}$
TabDDPM	3.3$_{\pm 1.2}$	17.9$_{\pm 19.8}$
TabSyn (org.)	17.9$_{\pm 23.6}$	0.9$_{\pm 0.9}$
TabDiff (org.)	20.6$_{\pm 26.1}$	3.8$_{\pm 3.2}$
CDTD	3.7$_{\pm 1.1}$	1.7$_{\pm 1.6}$
XGB-DDPM	4.9$_{\pm 3.4}$	11.1$_{\pm 14.1}$

that XGB-DDPM performs on par with state-of-the-art neural baseline models.

Table 7: Main results (Table 1), now considering original default hyperparameters (org.) instead of a hyperparameter alignment strategy (Appendix A.2).

	C2ST	C2ST (test)	WD (num.)	JSD (cat.)	MI	α-Precision	β-Recall	MLE	DCR Share	MIA Lift
ARF	$37.1_{\pm 12.2}$	$32.4_{\pm 13.2}$	$1.1_{\pm 1.2}$	**$0.8_{\pm 0.7}$**	$5.3_{\pm 3.2}$	**$3.3_{\pm 2.5}$**	$15.2_{\pm 12.3}$	$6.0_{\pm 6.6}$	$3.1_{\pm 1.7}$	$4.8_{\pm 2.4}$
CTGAN (org.)	$46.4_{\pm 5.6}$	$44.7_{\pm 6.4}$	$3.4_{\pm 3.2}$	$16.7_{\pm 10.7}$	$10.8_{\pm 10.2}$	$22.7_{\pm 11.4}$	$49.0_{\pm 27.4}$	$26.5_{\pm 26.7}$	**$1.1_{\pm 1.1}$**	**$2.8_{\pm 2.4}$**
TVAE (org.)	$47.7_{\pm 2.6}$	$46.1_{\pm 3.6}$	$2.6_{\pm 2.7}$	$17.9_{\pm 10.7}$	$10.3_{\pm 9.4}$	$22.5_{\pm 23.2}$	$47.7_{\pm 23.5}$	$9.6_{\pm 4.7}$	$1.7_{\pm 1.9}$	$2.9_{\pm 2.5}$
ForestDiffusion	$41.1_{\pm 15.3}$	$38.9_{\pm 16.4}$	$1.2_{\pm 0.6}$	$2.8_{\pm 3.6}$	$12.4_{\pm 11.6}$	$10.5_{\pm 10.5}$	$47.9_{\pm 34.7}$	$10.9_{\pm 11.1}$	$2.4_{\pm 2.5}$	$3.0_{\pm 2.6}$
ForestFlow	$38.7_{\pm 17.7}$	$37.2_{\pm 16.7}$	$0.7_{\pm 0.7}$	$2.8_{\pm 3.6}$	$3.2_{\pm 2.7}$	$6.3_{\pm 3.2}$	$17.9_{\pm 19.8}$	$8.1_{\pm 7.7}$	$2.7_{\pm 3.0}$	$3.9_{\pm 3.6}$
TabDDPM	$18.5_{\pm 16.5}$	$13.8_{\pm 9.9}$	$1.0_{\pm 1.2}$	$1.9_{\pm 1.9}$	$2.8_{\pm 2.5}$	$3.5_{\pm 2.9}$	$6.2_{\pm 4.7}$	$4.8_{\pm 5.4}$	$4.2_{\pm 5.1}$	$10.8_{\pm 13.5}$
TabSyn (org.)	$28.5_{\pm 10.8}$	$20.0_{\pm 7.7}$	$1.3_{\pm 1.4}$	$3.1_{\pm 3.0}$	$6.3_{\pm 8.4}$	$6.3_{\pm 6.8}$	$14.0_{\pm 11.5}$	$13.2_{\pm 14.8}$	$2.5_{\pm 3.5}$	$5.1_{\pm 2.8}$
TabDiff (org.)	$21.0_{\pm 12.7}$	$13.4_{\pm 7.1}$	$1.1_{\pm 1.4}$	$1.0_{\pm 0.9}$	**$2.1_{\pm 1.9}$**	$4.6_{\pm 5.4}$	$9.9_{\pm 7.9}$	$3.8_{\pm 3.9}$	$2.8_{\pm 3.7}$	$5.6_{\pm 3.2}$
CDTD	**$17.8_{\pm 14.4}$**	$13.4_{\pm 7.1}$	**$0.6_{\pm 0.5}$**	$1.1_{\pm 0.7}$	$2.2_{\pm 2.5}$	$3.4_{\pm 1.8}$	**$2.9_{\pm 3.5}$**	**$3.4_{\pm 3.3}$**	$6.7_{\pm 5.5}$	$12.6_{\pm 11.2}$
XGB-DDPM	$18.2_{\pm 14.4}$	**$12.3_{\pm 7.3}$**	**$0.6_{\pm 0.4}$**	$1.4_{\pm 1.1}$	$4.4_{\pm 6.2}$	$5.2_{\pm 3.2}$	$7.6_{\pm 6.0}$	$4.1_{\pm 4.5}$	$2.2_{\pm 2.8}$	$5.0_{\pm 2.7}$

B.2 Tradeoff between fidelity, overfitting, and membership disclosure.

Figure 4 shows the tradeoff between multivariate fidelity (C2ST), overfitting (DCR Share), and membership disclosure risk (MIA Lift).

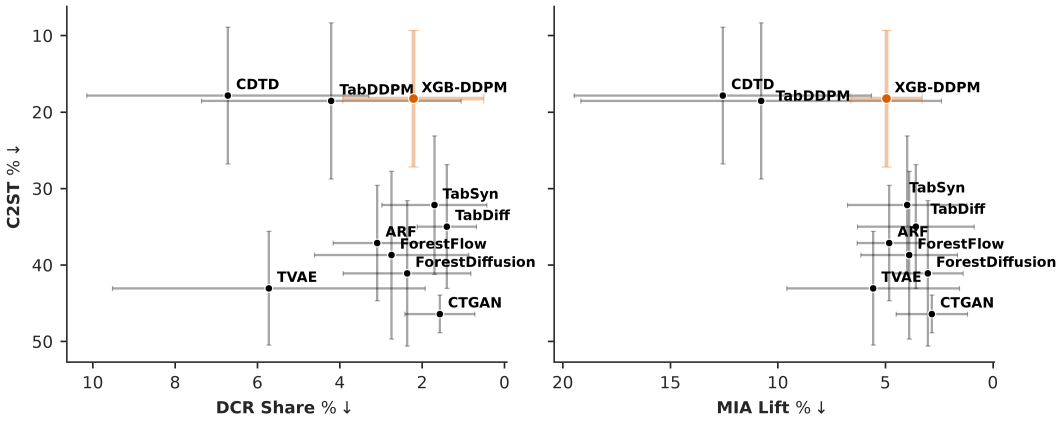

Figure 4: **Scatterplot of metric error rates (avg. $\pm$ 95% CI).** XGB-DDPM achieves high fidelity without increasing overfitting or membership disclosure risk compared to high-fidelity baselines.

B.3 Impact of diffusion objectives, samplers, and noise schedule

Figure 5 shows the impact of the diffusion objectives, samplers, and noise schedule, in a three-way interaction plot.

B.4 Impact of training models per timestep and target label

Figure 6 shows the impact of training models per timestep and target label.

B.5 Resource usage scaling with data size

Figure 7 shows the impact of increasing observations, features, and target classes on peak memory usage and runtime of XGB-DDPM versus ForestDiffusion.

B.6 Impact of learning rate and boosting rounds

Figure 8 shows the interplay between boosting rounds and learning rate in the XGBoost models. Dynamically lowering the learning rate for higher boosting rounds can be beneficial for fidelity.

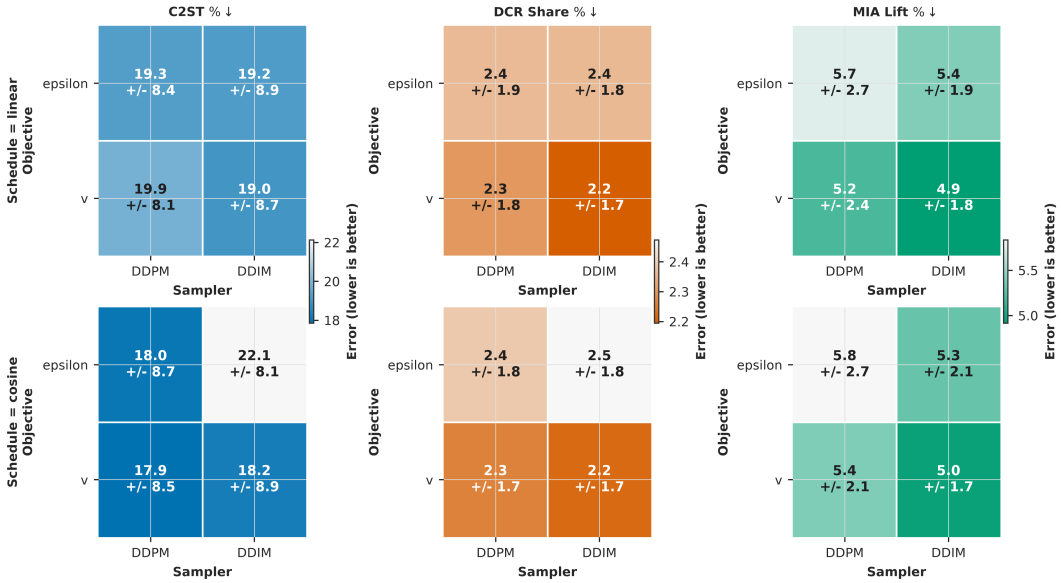

Figure 5: **Interaction between diffusion objectives, samplers, and noise schedule.** Overall, a cosine schedule, DDIM sampler, and v-objective yield balanced results across C2ST, DCR Share, and MIA Lift.

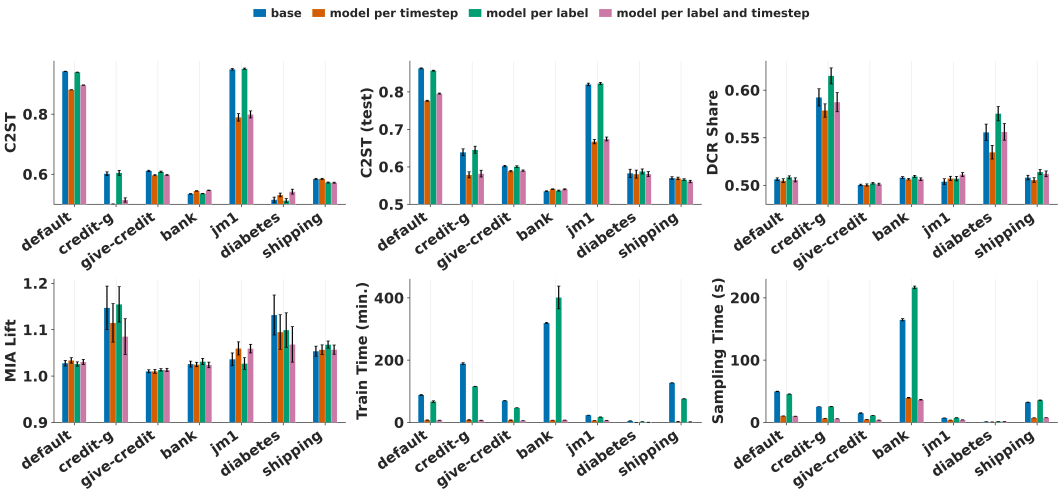

Figure 6: **Effect of training a single model per feature (base), or also per timestep, target label, or both.** We consider only classification datasets since regression datasets do not condition on the target. We find that especially training models per timestep yields high fidelity and low training times.

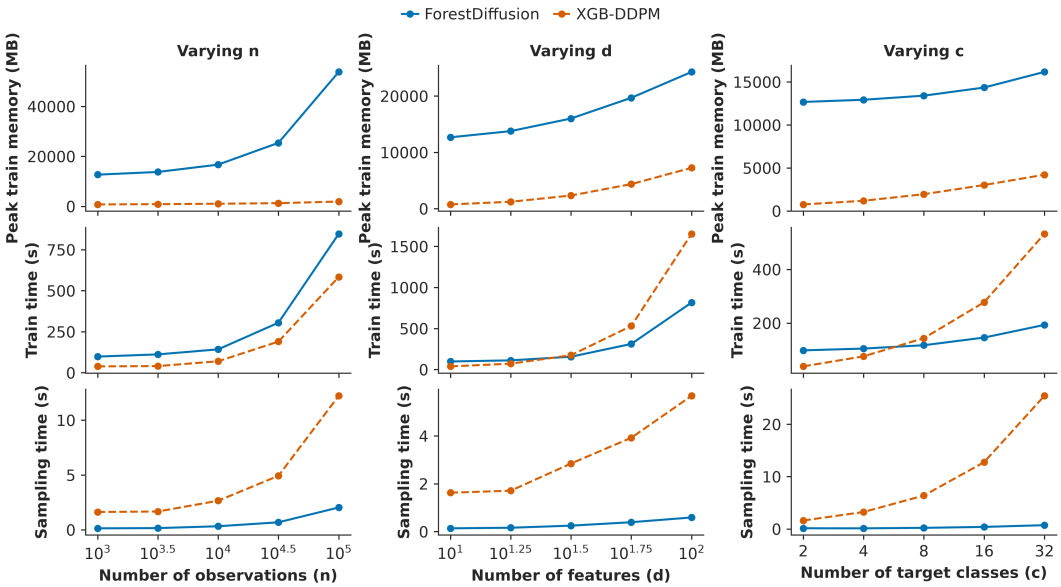

Figure 7: **Resource usage scaling with increasing n (observations), d (features), and target classes (c).** XGB-DDPM provides substantial memory benefits, and training time benefits for large n. For large d and c, ForestDiffusion is more efficient. ForestDiffusion is also more efficient at sampling.

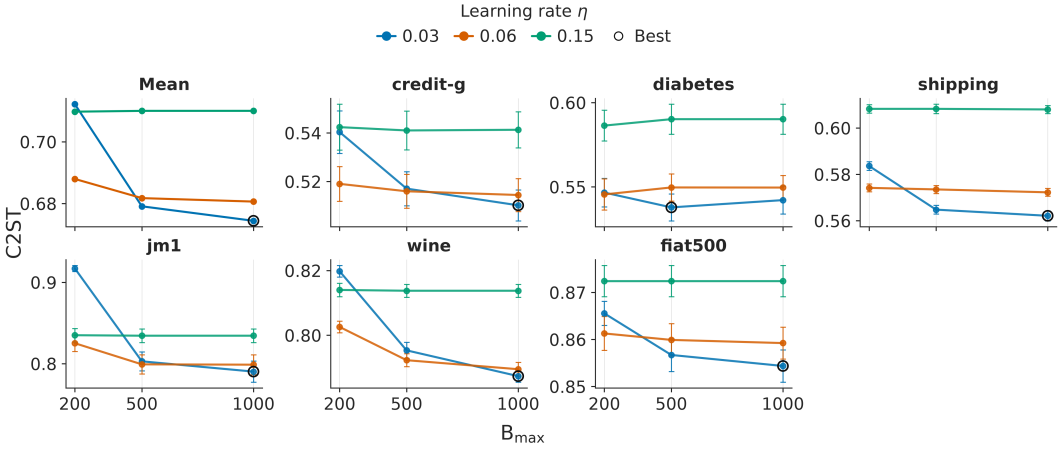

Figure 8: **Interaction between η and B_{max} for the small- and medium-sized experimental datasets**. High B_{max} and low η provide best results; for low B_{max} it can be beneficial to set higher η.

B.7 DETAILED MAIN RESULTS

Table 8 until 23 provide the detailed results which are aggregated in Table 1. These can be used to further investigate the results across each dataset.

Table 8: **C2ST** AUC for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined. Any AUC < 0.5 is similarly considered as no distinguishability.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.942_{\pm 0.009}$	$0.999_{\pm 0.000}$	$0.869_{\pm 0.009}$	$0.947_{\pm 0.002}$	$0.764_{\pm 0.006}$	$0.999_{\pm 0.000}$	$0.716_{\pm 0.004}$	$0.987_{\pm 0.014}$	$0.810_{\pm 0.026}$	$0.678_{\pm 0.006}$
CTGAN	$0.941_{\pm 0.008}$	$1.000_{\pm 0.000}$	$0.914_{\pm 0.018}$	$0.995_{\pm 0.001}$	$0.995_{\pm 0.001}$	$0.999_{\pm 0.000}$	$0.958_{\pm 0.021}$	$1.000_{\pm 0.000}$	$0.943_{\pm 0.014}$	$0.893_{\pm 0.006}$
CTGAN (org.)	$0.986_{\pm 0.003}$	$1.000_{\pm 0.000}$	$0.952_{\pm 0.005}$	$0.994_{\pm 0.000}$	$0.993_{\pm 0.001}$	$0.998_{\pm 0.001}$	$0.841_{\pm 0.005}$	$1.000_{\pm 0.000}$	$0.986_{\pm 0.005}$	$0.885_{\pm 0.008}$
TVAE	$0.940_{\pm 0.005}$	$1.000_{\pm 0.000}$	$0.750_{\pm 0.044}$	$0.973_{\pm 0.004}$	$0.997_{\pm 0.001}$	$0.999_{\pm 0.000}$	$0.987_{\pm 0.004}$	$1.000_{\pm 0.000}$	$0.667_{\pm 0.030}$	$0.990_{\pm 0.008}$
TVAE (org.)	$0.971_{\pm 0.003}$	$1.000_{\pm 0.000}$	$0.993_{\pm 0.001}$	$0.987_{\pm 0.006}$	$0.965_{\pm 0.001}$	$0.999_{\pm 0.000}$	$0.923_{\pm 0.009}$	$1.000_{\pm 0.000}$	$0.944_{\pm 0.008}$	$0.990_{\pm 0.001}$
ForestDiffusion	$0.910_{\pm 0.006}$	$1.000_{\pm 0.000}$	$0.969_{\pm 0.004}$	$0.995_{\pm 0.001}$	$1.000_{\pm 0.000}$	$0.999_{\pm 0.000}$	$0.979_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.677_{\pm 0.018}$	$0.579_{\pm 0.005}$
ForestFlow	$0.909_{\pm 0.007}$	$0.999_{\pm 0.000}$	$0.962_{\pm 0.004}$	$0.962_{\pm 0.003}$	$1.000_{\pm 0.000}$	$0.999_{\pm 0.000}$	$0.933_{\pm 0.000}$	$0.993_{\pm 0.010}$	$0.582_{\pm 0.017}$	$\mathbf{0.532_{\pm 0.005}}$
TabDDPM	$0.874_{\pm 0.011}$	$0.945_{\pm 0.007}$	$\mathbf{0.482_{\pm 0.025}}$	$0.799_{\pm 0.005}$	$\mathbf{0.546_{\pm 0.004}}$	$0.799_{\pm 0.006}$	$0.538_{\pm 0.002}$	$\mathbf{0.733_{\pm 0.052}}$	$\mathbf{0.419_{\pm 0.018}}$	$0.616_{\pm 0.009}$
TabSyn	$0.927_{\pm 0.014}$	$0.931_{\pm 0.008}$	$0.952_{\pm 0.017}$	$0.803_{\pm 0.027}$	$0.586_{\pm 0.005}$	$0.864_{\pm 0.007}$	$0.576_{\pm 0.006}$	$0.995_{\pm 0.007}$	$0.827_{\pm 0.032}$	$0.757_{\pm 0.064}$
TabSyn (org.)	$0.899_{\pm 0.008}$	$0.868_{\pm 0.012}$	$0.780_{\pm 0.031}$	$0.838_{\pm 0.117}$	$0.828_{\pm 0.197}$	$0.840_{\pm 0.007}$	$0.691_{\pm 0.201}$	$0.877_{\pm 0.028}$	$0.618_{\pm 0.020}$	$0.608_{\pm 0.015}$
TabDiff	$0.939_{\pm 0.005}$	$0.964_{\pm 0.014}$	$0.845_{\pm 0.048}$	$0.708_{\pm 0.009}$	$0.930_{\pm 0.034}$	$0.861_{\pm 0.030}$	$0.602_{\pm 0.090}$	$0.999_{\pm 0.000}$	$0.925_{\pm 0.014}$	$0.723_{\pm 0.018}$
TabDiff (org.)	$0.894_{\pm 0.009}$	$0.868_{\pm 0.094}$	$0.709_{\pm 0.043}$	$\mathbf{0.665_{\pm 0.010}}$	$0.553_{\pm 0.004}$	$0.796_{\pm 0.008}$	$\mathbf{0.522_{\pm 0.004}}$	$0.767_{\pm 0.043}$	$0.734_{\pm 0.069}$	$0.593_{\pm 0.084}$
CDTD	$\mathbf{0.812_{\pm 0.014}}$	$\mathbf{0.842_{\pm 0.004}}$	$\mathbf{0.481_{\pm 0.026}}$	$0.759_{\pm 0.002}$	$0.597_{\pm 0.011}$	$\mathbf{0.731_{\pm 0.006}}$	$0.549_{\pm 0.003}$	$0.886_{\pm 0.013}$	$\mathbf{0.489_{\pm 0.024}}$	$0.600_{\pm 0.006}$
XGB-DDPM	$0.854_{\pm 0.010}$	$0.897_{\pm 0.002}$	$0.514_{\pm 0.020}$	$0.706_{\pm 0.003}$	$0.598_{\pm 0.004}$	$0.792_{\pm 0.006}$	$0.546_{\pm 0.003}$	$0.799_{\pm 0.033}$	$0.542_{\pm 0.023}$	$0.572_{\pm 0.005}$

Table 9: **C2ST (test)** AUC for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined. Any AUC < 0.5 is similarly considered as no distinguishability.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.826_{\pm 0.023}$	$0.997_{\pm 0.000}$	$0.714_{\pm 0.026}$	$0.925_{\pm 0.004}$	$0.758_{\pm 0.005}$	$0.981_{\pm 0.002}$	$0.703_{\pm 0.003}$	$0.963_{\pm 0.034}$	$0.719_{\pm 0.028}$	$0.648_{\pm 0.008}$
CTGAN	$0.853_{\pm 0.015}$	$1.000_{\pm 0.000}$	$0.837_{\pm 0.033}$	$0.989_{\pm 0.002}$	$0.994_{\pm 0.001}$	$0.977_{\pm 0.003}$	$0.951_{\pm 0.024}$	$0.999_{\pm 0.001}$	$0.911_{\pm 0.017}$	$0.865_{\pm 0.008}$
CTGAN (org.)	$0.964_{\pm 0.005}$	$0.999_{\pm 0.000}$	$0.898_{\pm 0.011}$	$0.987_{\pm 0.001}$	$0.993_{\pm 0.002}$	$0.978_{\pm 0.004}$	$0.822_{\pm 0.008}$	$1.000_{\pm 0.000}$	$0.969_{\pm 0.009}$	$0.857_{\pm 0.010}$
TVAE	$0.825_{\pm 0.023}$	$0.999_{\pm 0.000}$	$0.698_{\pm 0.047}$	$0.948_{\pm 0.006}$	$0.997_{\pm 0.001}$	$0.980_{\pm 0.003}$	$0.984_{\pm 0.006}$	$0.999_{\pm 0.001}$	$0.665_{\pm 0.020}$	$0.984_{\pm 0.014}$
TVAE (org.)	$0.906_{\pm 0.014}$	$0.999_{\pm 0.000}$	$0.981_{\pm 0.006}$	$0.976_{\pm 0.011}$	$0.963_{\pm 0.001}$	$0.975_{\pm 0.005}$	$0.913_{\pm 0.009}$	$0.998_{\pm 0.001}$	$0.914_{\pm 0.014}$	$0.985_{\pm 0.002}$
ForestDiffusion	$0.791_{\pm 0.014}$	$0.997_{\pm 0.000}$	$0.953_{\pm 0.009}$	$0.992_{\pm 0.001}$	$1.000_{\pm 0.000}$	$0.973_{\pm 0.003}$	$0.977_{\pm 0.001}$	$0.998_{\pm 0.000}$	$0.644_{\pm 0.034}$	$0.562_{\pm 0.008}$
ForestFlow	$0.756_{\pm 0.014}$	$0.996_{\pm 0.000}$	$0.945_{\pm 0.007}$	$0.952_{\pm 0.004}$	$1.000_{\pm 0.000}$	$0.985_{\pm 0.002}$	$0.929_{\pm 0.002}$	$0.981_{\pm 0.024}$	$0.626_{\pm 0.021}$	$\mathbf{0.547_{\pm 0.007}}$
TabDDPM	$0.667_{\pm 0.021}$	$0.877_{\pm 0.016}$	$0.604_{\pm 0.027}$	$0.715_{\pm 0.006}$	$\mathbf{0.544_{\pm 0.004}}$	$0.633_{\pm 0.009}$	$0.537_{\pm 0.004}$	$\mathbf{0.617_{\pm 0.015}}$	$0.583_{\pm 0.024}$	$0.600_{\pm 0.009}$
TabSyn	$0.761_{\pm 0.048}$	$0.835_{\pm 0.013}$	$0.908_{\pm 0.028}$	$0.714_{\pm 0.029}$	$0.579_{\pm 0.005}$	$0.681_{\pm 0.015}$	$0.553_{\pm 0.003}$	$0.985_{\pm 0.018}$	$0.776_{\pm 0.045}$	$0.718_{\pm 0.070}$
TabSyn (org.)	$0.690_{\pm 0.014}$	$\mathbf{0.729_{\pm 0.010}}$	$0.751_{\pm 0.048}$	$0.782_{\pm 0.157}$	$0.827_{\pm 0.196}$	$0.637_{\pm 0.012}$	$0.684_{\pm 0.204}$	$0.706_{\pm 0.022}$	$0.607_{\pm 0.028}$	$0.582_{\pm 0.015}$
TabDiff	$0.829_{\pm 0.014}$	$0.928_{\pm 0.029}$	$0.761_{\pm 0.039}$	$0.638_{\pm 0.009}$	$0.928_{\pm 0.035}$	$0.717_{\pm 0.055}$	$0.594_{\pm 0.089}$	$0.997_{\pm 0.001}$	$0.877_{\pm 0.023}$	$0.716_{\pm 0.011}$
TabDiff (org.)	$0.681_{\pm 0.019}$	$0.780_{\pm 0.155}$	$0.667_{\pm 0.035}$	$\mathbf{0.609_{\pm 0.007}}$	$0.550_{\pm 0.004}$	$\mathbf{0.622_{\pm 0.010}}$	$\mathbf{0.530_{\pm 0.004}}$	$0.629_{\pm 0.016}$	$0.672_{\pm 0.038}$	$0.598_{\pm 0.080}$
CDTD	$\mathbf{0.635_{\pm 0.021}}$	$0.734_{\pm 0.004}$	$\mathbf{0.582_{\pm 0.022}}$	$0.677_{\pm 0.004}$	$0.591_{\pm 0.011}$	$0.625_{\pm 0.009}$	$0.545_{\pm 0.005}$	$0.765_{\pm 0.013}$	$0.585_{\pm 0.023}$	$0.602_{\pm 0.010}$
XGB-DDPM	$0.636_{\pm 0.021}$	$0.796_{\pm 0.004}$	$\mathbf{0.582_{\pm 0.025}}$	$0.630_{\pm 0.003}$	$0.590_{\pm 0.004}$	$0.636_{\pm 0.010}$	$0.540_{\pm 0.004}$	$0.674_{\pm 0.014}$	$\mathbf{0.581_{\pm 0.019}}$	$0.561_{\pm 0.009}$

Table 10: **WD (num.)** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.020_{\pm 0.002}$	$0.006_{\pm 0.000}$	$0.039_{\pm 0.003}$	$\mathbf{0.001_{\pm 0.000}}$	$0.002_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.004_{\pm 0.000}$	$\mathbf{0.001_{\pm 0.000}}$	$0.020_{\pm 0.001}$	$0.014_{\pm 0.001}$
CTGAN	$0.053_{\pm 0.007}$	$0.007_{\pm 0.001}$	$0.063_{\pm 0.021}$	$0.007_{\pm 0.001}$	$0.007_{\pm 0.001}$	$0.024_{\pm 0.003}$	$0.013_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.039_{\pm 0.009}$	$0.041_{\pm 0.002}$
CTGAN (org.)	$0.058_{\pm 0.004}$	$0.006_{\pm 0.001}$	$0.091_{\pm 0.007}$	$0.007_{\pm 0.000}$	$0.005_{\pm 0.001}$	$0.032_{\pm 0.001}$	$0.009_{\pm 0.001}$	$0.006_{\pm 0.001}$	$0.072_{\pm 0.013}$	$0.053_{\pm 0.008}$
TVAE	$0.039_{\pm 0.023}$	$0.014_{\pm 0.001}$	$0.032_{\pm 0.004}$	$0.007_{\pm 0.001}$	$0.015_{\pm 0.002}$	$0.019_{\pm 0.005}$	$0.019_{\pm 0.003}$	$0.005_{\pm 0.001}$	$0.027_{\pm 0.002}$	$0.106_{\pm 0.089}$
TVAE (org.)	$0.035_{\pm 0.002}$	$0.009_{\pm 0.003}$	$0.086_{\pm 0.006}$	$0.008_{\pm 0.001}$	$0.007_{\pm 0.000}$	$0.014_{\pm 0.002}$	$0.009_{\pm 0.001}$	$0.003_{\pm 0.000}$	$0.042_{\pm 0.002}$	$0.047_{\pm 0.003}$
ForestDiffusion	$0.014_{\pm 0.001}$	$0.012_{\pm 0.000}$	$0.027_{\pm 0.002}$	$0.008_{\pm 0.000}$	$0.013_{\pm 0.002}$	$0.006_{\pm 0.000}$	$0.011_{\pm 0.000}$	$0.005_{\pm 0.000}$	$0.013_{\pm 0.002}$	$0.006_{\pm 0.001}$
ForestFlow	$0.009_{\pm 0.001}$	$0.003_{\pm 0.000}$	$0.024_{\pm 0.001}$	$0.004_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.012_{\pm 0.001}$	$0.006_{\pm 0.001}$
TabDDPM	$0.019_{\pm 0.011}$	$0.039_{\pm 0.015}$	$0.015_{\pm 0.003}$	$0.004_{\pm 0.001}$	$\mathbf{0.001_{\pm 0.000}}$	$0.006_{\pm 0.001}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.008_{\pm 0.001}}$	$0.011_{\pm 0.001}$
TabSyn	$0.016_{\pm 0.005}$	$0.004_{\pm 0.000}$	$0.116_{\pm 0.032}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$0.005_{\pm 0.000}$	$\mathbf{0.001_{\pm 0.000}}$	$0.016_{\pm 0.006}$	$0.041_{\pm 0.009}$	$0.023_{\pm 0.008}$
TabSyn (org.)	$0.009_{\pm 0.001}$	$\mathbf{0.002_{\pm 0.000}}$	$0.047_{\pm 0.012}$	$0.014_{\pm 0.019}$	$0.011_{\pm 0.008}$	$0.004_{\pm 0.000}$	$0.025_{\pm 0.034}$	$\mathbf{0.001_{\pm 0.000}}$	$0.013_{\pm 0.001}$	$0.008_{\pm 0.002}$
TabDiff	$0.017_{\pm 0.002}$	$0.008_{\pm 0.001}$	$0.065_{\pm 0.005}$	$\mathbf{0.001_{\pm 0.000}}$	$0.017_{\pm 0.007}$	$0.005_{\pm 0.002}$	$0.005_{\pm 0.006}$	$0.026_{\pm 0.002}$	$0.067_{\pm 0.003}$	$0.027_{\pm 0.002}$
TabDiff (org.)	$0.011_{\pm 0.002}$	$0.009_{\pm 0.011}$	$0.044_{\pm 0.007}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.003_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$0.026_{\pm 0.008}$	$0.011_{\pm 0.010}$
CDTD	$\mathbf{0.007_{\pm 0.002}}$	$0.003_{\pm 0.000}$	$0.014_{\pm 0.002}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$0.004_{\pm 0.000}$	$0.002_{\pm 0.000}$	$\mathbf{0.001_{\pm 0.000}}$	$0.011_{\pm 0.002}$	$0.013_{\pm 0.001}$
XGB-DDPM	$0.010_{\pm 0.002}$	$0.004_{\pm 0.000}$	$\mathbf{0.013_{\pm 0.002}}$	$0.003_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.006_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.011_{\pm 0.001}$	$\mathbf{0.005_{\pm 0.001}}$

Table 11: **WD (num., test)** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.027_{\pm 0.005}$	$0.007_{\pm 0.001}$	$0.047_{\pm 0.004}$	$0.003_{\pm 0.000}$	$0.002_{\pm 0.000}$	$0.008_{\pm 0.001}$	$0.006_{\pm 0.000}$	$\mathbf{0.002_{\pm 0.000}}$	$0.036_{\pm 0.002}$	$0.014_{\pm 0.001}$
CTGAN	$0.057_{\pm 0.005}$	$0.008_{\pm 0.001}$	$0.069_{\pm 0.016}$	$0.010_{\pm 0.002}$	$0.007_{\pm 0.001}$	$0.031_{\pm 0.002}$	$0.015_{\pm 0.001}$	$0.008_{\pm 0.002}$	$0.061_{\pm 0.007}$	$0.040_{\pm 0.002}$
CTGAN (org.)	$0.063_{\pm 0.005}$	$0.008_{\pm 0.001}$	$0.104_{\pm 0.008}$	$0.009_{\pm 0.002}$	$0.006_{\pm 0.001}$	$0.037_{\pm 0.004}$	$0.010_{\pm 0.001}$	$0.008_{\pm 0.002}$	$0.089_{\pm 0.013}$	$0.053_{\pm 0.010}$
TVAE	$0.045_{\pm 0.027}$	$0.018_{\pm 0.002}$	$0.039_{\pm 0.004}$	$0.010_{\pm 0.001}$	$0.017_{\pm 0.001}$	$0.022_{\pm 0.005}$	$0.023_{\pm 0.003}$	$0.007_{\pm 0.001}$	$0.041_{\pm 0.008}$	$0.104_{\pm 0.089}$
TVAE (org.)	$0.036_{\pm 0.002}$	$0.011_{\pm 0.003}$	$0.099_{\pm 0.007}$	$0.011_{\pm 0.002}$	$0.008_{\pm 0.000}$	$0.018_{\pm 0.004}$	$0.010_{\pm 0.001}$	$0.004_{\pm 0.000}$	$0.049_{\pm 0.005}$	$0.047_{\pm 0.005}$
ForestDiffusion	$0.021_{\pm 0.004}$	$0.017_{\pm 0.002}$	$0.040_{\pm 0.004}$	$0.017_{\pm 0.007}$	$0.015_{\pm 0.002}$	$0.010_{\pm 0.001}$	$0.024_{\pm 0.002}$	$0.008_{\pm 0.002}$	$0.028_{\pm 0.004}$	$0.009_{\pm 0.001}$
ForestFlow	$0.014_{\pm 0.002}$	$0.005_{\pm 0.000}$	$0.039_{\pm 0.003}$	$0.006_{\pm 0.001}$	$0.002_{\pm 0.000}$	$0.009_{\pm 0.001}$	$0.004_{\pm 0.000}$	$0.003_{\pm 0.000}$	$0.028_{\pm 0.004}$	$0.008_{\pm 0.001}$
TabDDPM	$0.026_{\pm 0.013}$	$0.059_{\pm 0.033}$	$0.028_{\pm 0.006}$	$0.008_{\pm 0.002}$	$\mathbf{0.001_{\pm 0.000}}$	$0.010_{\pm 0.001}$	$\mathbf{0.002_{\pm 0.000}}$	$\mathbf{0.002_{\pm 0.000}}$	$0.027_{\pm 0.003}$	$0.012_{\pm 0.002}$
TabSyn	$0.020_{\pm 0.005}$	$0.005_{\pm 0.000}$	$0.119_{\pm 0.030}$	$\mathbf{0.002_{\pm 0.000}}$	$0.002_{\pm 0.000}$	$0.008_{\pm 0.001}$	$\mathbf{0.002_{\pm 0.000}}$	$0.022_{\pm 0.008}$	$0.061_{\pm 0.018}$	$0.024_{\pm 0.008}$
TabSyn (org.)	$\mathbf{0.013_{\pm 0.002}}$	$\mathbf{0.003_{\pm 0.000}}$	$0.057_{\pm 0.014}$	$0.041_{\pm 0.056}$	$0.014_{\pm 0.010}$	$\mathbf{0.007_{\pm 0.001}}$	$0.032_{\pm 0.043}$	$\mathbf{0.002_{\pm 0.000}}$	$0.030_{\pm 0.004}$	$0.010_{\pm 0.001}$
TabDiff	$0.020_{\pm 0.002}$	$0.012_{\pm 0.002}$	$0.068_{\pm 0.009}$	$\mathbf{0.002_{\pm 0.000}}$	$0.020_{\pm 0.009}$	$0.009_{\pm 0.002}$	$0.006_{\pm 0.007}$	$0.036_{\pm 0.013}$	$0.093_{\pm 0.006}$	$0.029_{\pm 0.002}$
TabDiff (org.)	$0.015_{\pm 0.002}$	$0.015_{\pm 0.017}$	$0.049_{\pm 0.005}$	$\mathbf{0.002_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.007_{\pm 0.001}}$	$\mathbf{0.002_{\pm 0.000}}$	$\mathbf{0.002_{\pm 0.001}}$	$0.044_{\pm 0.010}$	$0.014_{\pm 0.010}$
CDTD	$\mathbf{0.013_{\pm 0.002}}$	$0.004_{\pm 0.000}$	$\mathbf{0.026_{\pm 0.006}}$	$\mathbf{0.002_{\pm 0.000}}$	$\mathbf{0.001_{\pm 0.000}}$	$\mathbf{0.007_{\pm 0.001}}$	$0.003_{\pm 0.000}$	$\mathbf{0.002_{\pm 0.000}}$	$\mathbf{0.026_{\pm 0.004}}$	$0.014_{\pm 0.001}$
XGB-DDPM	$0.014_{\pm 0.002}$	$0.006_{\pm 0.000}$	$0.028_{\pm 0.005}$	$0.004_{\pm 0.001}$	$0.002_{\pm 0.000}$	$0.009_{\pm 0.001}$	$0.004_{\pm 0.000}$	$0.004_{\pm 0.000}$	$0.028_{\pm 0.003}$	$\mathbf{0.007_{\pm 0.001}}$

Table 12: **JSD (cat.)** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.015_{\pm 0.007}$	$\mathbf{0.004_{\pm 0.001}}$	$\mathbf{0.023_{\pm 0.003}}$	$\mathbf{0.004_{\pm 0.001}}$	$\mathbf{0.002_{\pm 0.002}}$	$\mathbf{0.005_{\pm 0.004}}$	$\mathbf{0.003_{\pm 0.000}}$	$\mathbf{0.003_{\pm 0.003}}$	$0.013_{\pm 0.009}$	$\mathbf{0.006_{\pm 0.002}}$
CTGAN	$0.296_{\pm 0.009}$	$0.109_{\pm 0.019}$	$0.141_{\pm 0.015}$	$0.154_{\pm 0.011}$	$0.377_{\pm 0.002}$	$0.194_{\pm 0.004}$	$0.103_{\pm 0.006}$	$0.241_{\pm 0.004}$	$0.107_{\pm 0.021}$	$0.107_{\pm 0.024}$
CTGAN (org.)	$0.296_{\pm 0.009}$	$0.130_{\pm 0.004}$	$0.049_{\pm 0.009}$	$0.124_{\pm 0.013}$	$0.377_{\pm 0.002}$	$0.194_{\pm 0.004}$	$0.076_{\pm 0.018}$	$0.241_{\pm 0.004}$	$0.107_{\pm 0.021}$	$0.080_{\pm 0.005}$
TVAE	$0.116_{\pm 0.043}$	$0.201_{\pm 0.012}$	$0.062_{\pm 0.016}$	$0.147_{\pm 0.018}$	$0.104_{\pm 0.066}$	$0.027_{\pm 0.015}$	$0.194_{\pm 0.020}$	$0.193_{\pm 0.058}$	$0.039_{\pm 0.025}$	$0.351_{\pm 0.139}$
TVAE (org.)	$0.343_{\pm 0.021}$	$0.188_{\pm 0.016}$	$0.300_{\pm 0.007}$	$0.145_{\pm 0.008}$	$0.107_{\pm 0.048}$	$0.017_{\pm 0.016}$	$0.115_{\pm 0.006}$	$0.232_{\pm 0.025}$	$0.069_{\pm 0.024}$	$0.278_{\pm 0.006}$
ForestDiffusion	$0.022_{\pm 0.010}$	$0.012_{\pm 0.001}$	$0.113_{\pm 0.004}$	$0.019_{\pm 0.001}$	$\mathbf{0.002_{\pm 0.002}}$	$0.009_{\pm 0.007}$	$0.074_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.019_{\pm 0.014}$	$0.008_{\pm 0.002}$
ForestFlow	$0.017_{\pm 0.008}$	$0.012_{\pm 0.001}$	$0.111_{\pm 0.002}$	$0.028_{\pm 0.002}$	$\mathbf{0.002_{\pm 0.002}}$	$0.008_{\pm 0.005}$	$0.075_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.019_{\pm 0.014}$	$0.008_{\pm 0.001}$
TabDDPM	$0.059_{\pm 0.012}$	$0.030_{\pm 0.006}$	$0.040_{\pm 0.005}$	$0.008_{\pm 0.001}$	$\mathbf{0.002_{\pm 0.001}}$	$\mathbf{0.005_{\pm 0.003}}$	$0.008_{\pm 0.001}$	$0.004_{\pm 0.002}$	$0.016_{\pm 0.011}$	$0.017_{\pm 0.003}$
TabSyn	$0.019_{\pm 0.010}$	$0.025_{\pm 0.017}$	$0.089_{\pm 0.009}$	$0.007_{\pm 0.002}$	$0.018_{\pm 0.004}$	$\mathbf{0.005_{\pm 0.005}}$	$0.005_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.022_{\pm 0.020}$	$0.035_{\pm 0.028}$
TabSyn (org.)	$0.017_{\pm 0.009}$	$0.015_{\pm 0.006}$	$0.089_{\pm 0.005}$	$0.069_{\pm 0.091}$	$0.047_{\pm 0.035}$	$\mathbf{0.005_{\pm 0.005}}$	$0.042_{\pm 0.054}$	$0.004_{\pm 0.003}$	$0.013_{\pm 0.011}$	$\mathbf{0.006_{\pm 0.001}}$
TabDiff	$0.013_{\pm 0.008}$	$0.015_{\pm 0.002}$	$\mathbf{0.023_{\pm 0.003}}$	$0.008_{\pm 0.001}$	$0.036_{\pm 0.008}$	$0.006_{\pm 0.004}$	$\mathbf{0.003_{\pm 0.001}}$	$\mathbf{0.003_{\pm 0.002}}$	$0.014_{\pm 0.009}$	$\mathbf{0.006_{\pm 0.001}}$
TabDiff (org.)	$\mathbf{0.011_{\pm 0.006}}$	$0.016_{\pm 0.008}$	$0.033_{\pm 0.004}$	$0.007_{\pm 0.001}$	$0.004_{\pm 0.002}$	$0.006_{\pm 0.004}$	$0.005_{\pm 0.001}$	$\mathbf{0.003_{\pm 0.002}}$	$\mathbf{0.011_{\pm 0.008}}$	$\mathbf{0.006_{\pm 0.001}}$
CDTD	$0.018_{\pm 0.007}$	$0.021_{\pm 0.001}$	$0.024_{\pm 0.003}$	$0.005_{\pm 0.001}$	$0.005_{\pm 0.002}$	$\mathbf{0.005_{\pm 0.004}}$	$0.008_{\pm 0.001}$	$0.007_{\pm 0.004}$	$0.015_{\pm 0.010}$	$0.007_{\pm 0.001}$
XGB-DDPM	$0.021_{\pm 0.010}$	$0.008_{\pm 0.001}$	$0.037_{\pm 0.004}$	$0.014_{\pm 0.001}$	$\mathbf{0.002_{\pm 0.001}}$	$0.024_{\pm 0.005}$	$0.007_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.014_{\pm 0.009}$	$\mathbf{0.006_{\pm 0.001}}$

Table 13: **JSD (cat., test)** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.046_{\pm 0.023}$	$\mathbf{0.008_{\pm 0.001}}$	$\mathbf{0.055_{\pm 0.005}}$	$\mathbf{0.012_{\pm 0.002}}$	$0.002_{\pm 0.001}$	$0.008_{\pm 0.006}$	$\mathbf{0.007_{\pm 0.001}}$	$\mathbf{0.003_{\pm 0.003}}$	$0.012_{\pm 0.009}$	$0.013_{\pm 0.003}$
CTGAN	$0.306_{\pm 0.011}$	$0.109_{\pm 0.018}$	$0.144_{\pm 0.018}$	$0.157_{\pm 0.011}$	$0.377_{\pm 0.001}$	$0.201_{\pm 0.006}$	$0.104_{\pm 0.005}$	$0.242_{\pm 0.004}$	$0.105_{\pm 0.021}$	$0.109_{\pm 0.023}$
CTGAN (org.)	$0.306_{\pm 0.011}$	$0.131_{\pm 0.004}$	$0.070_{\pm 0.009}$	$0.126_{\pm 0.015}$	$0.377_{\pm 0.001}$	$0.201_{\pm 0.006}$	$0.078_{\pm 0.018}$	$0.242_{\pm 0.004}$	$0.106_{\pm 0.021}$	$0.078_{\pm 0.005}$
TVAE	$0.127_{\pm 0.074}$	$0.204_{\pm 0.012}$	$0.083_{\pm 0.014}$	$0.145_{\pm 0.018}$	$0.104_{\pm 0.066}$	$0.023_{\pm 0.015}$	$0.194_{\pm 0.021}$	$0.193_{\pm 0.058}$	$0.041_{\pm 0.025}$	$0.353_{\pm 0.142}$
TVAE (org.)	$0.353_{\pm 0.027}$	$0.189_{\pm 0.017}$	$0.307_{\pm 0.007}$	$0.148_{\pm 0.012}$	$0.107_{\pm 0.048}$	$0.022_{\pm 0.016}$	$0.115_{\pm 0.006}$	$0.232_{\pm 0.025}$	$0.070_{\pm 0.024}$	$0.278_{\pm 0.004}$
ForestDiffusion	$0.049_{\pm 0.017}$	$0.014_{\pm 0.001}$	$0.127_{\pm 0.005}$	$0.023_{\pm 0.002}$	$0.002_{\pm 0.001}$	$0.015_{\pm 0.010}$	$0.077_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.018_{\pm 0.014}$	$0.013_{\pm 0.003}$
ForestFlow	$0.043_{\pm 0.019}$	$0.015_{\pm 0.001}$	$0.125_{\pm 0.006}$	$0.032_{\pm 0.002}$	$0.002_{\pm 0.001}$	$\mathbf{0.007_{\pm 0.005}}$	$0.078_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.018_{\pm 0.014}$	$\mathbf{0.012_{\pm 0.002}}$
TabDDPM	$0.072_{\pm 0.021}$	$0.030_{\pm 0.005}$	$0.065_{\pm 0.008}$	$\mathbf{0.012_{\pm 0.003}}$	$\mathbf{0.001_{\pm 0.001}}$	$0.010_{\pm 0.006}$	$0.011_{\pm 0.001}$	$0.004_{\pm 0.002}$	$0.016_{\pm 0.012}$	$0.021_{\pm 0.003}$
TabSyn	$0.043_{\pm 0.023}$	$0.027_{\pm 0.018}$	$0.103_{\pm 0.007}$	$0.013_{\pm 0.002}$	$0.018_{\pm 0.004}$	$0.008_{\pm 0.006}$	$0.009_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.023_{\pm 0.021}$	$0.039_{\pm 0.027}$
TabSyn (org.)	$0.044_{\pm 0.022}$	$0.017_{\pm 0.006}$	$0.103_{\pm 0.013}$	$0.072_{\pm 0.089}$	$0.047_{\pm 0.035}$	$0.009_{\pm 0.006}$	$0.046_{\pm 0.052}$	$0.004_{\pm 0.003}$	$0.013_{\pm 0.011}$	$\mathbf{0.012_{\pm 0.003}}$
TabDiff	$0.048_{\pm 0.019}$	$0.017_{\pm 0.002}$	$\mathbf{0.055_{\pm 0.006}}$	$0.013_{\pm 0.003}$	$0.037_{\pm 0.008}$	$0.010_{\pm 0.007}$	$0.008_{\pm 0.001}$	$\mathbf{0.003_{\pm 0.002}}$	$0.014_{\pm 0.010}$	$\mathbf{0.012_{\pm 0.003}}$
TabDiff (org.)	$0.044_{\pm 0.017}$	$0.018_{\pm 0.008}$	$0.063_{\pm 0.006}$	$0.014_{\pm 0.003}$	$0.004_{\pm 0.002}$	$0.009_{\pm 0.006}$	$0.009_{\pm 0.001}$	$\mathbf{0.003_{\pm 0.002}}$	$\mathbf{0.011_{\pm 0.008}}$	$0.013_{\pm 0.003}$
CDTD	$\mathbf{0.040_{\pm 0.018}}$	$0.021_{\pm 0.002}$	$0.058_{\pm 0.004}$	$\mathbf{0.012_{\pm 0.001}}$	$0.006_{\pm 0.002}$	$0.009_{\pm 0.007}$	$0.011_{\pm 0.002}$	$0.007_{\pm 0.004}$	$0.015_{\pm 0.010}$	$0.013_{\pm 0.004}$
XGB-DDPM	$0.045_{\pm 0.020}$	$0.011_{\pm 0.001}$	$0.062_{\pm 0.005}$	$0.016_{\pm 0.002}$	$\mathbf{0.001_{\pm 0.001}}$	$0.017_{\pm 0.007}$	$0.010_{\pm 0.001}$	$0.004_{\pm 0.003}$	$0.014_{\pm 0.009}$	$0.013_{\pm 0.003}$

Table 14: **MI** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.928_{\pm 0.003}$	$0.886_{\pm 0.002}$	$0.971_{\pm 0.001}$	$0.970_{\pm 0.003}$	$0.906_{\pm 0.022}$	$0.963_{\pm 0.002}$	$0.953_{\pm 0.007}$	$0.933_{\pm 0.006}$	$0.979_{\pm 0.002}$	$0.979_{\pm 0.001}$
CTGAN	$0.938_{\pm 0.003}$	$0.941_{\pm 0.015}$	$0.977_{\pm 0.002}$	$0.903_{\pm 0.007}$	$0.652_{\pm 0.018}$	$0.953_{\pm 0.016}$	$0.920_{\pm 0.006}$	$0.874_{\pm 0.007}$	$0.973_{\pm 0.004}$	$0.966_{\pm 0.001}$
CTGAN (org.)	$0.854_{\pm 0.004}$	$0.937_{\pm 0.002}$	$0.963_{\pm 0.001}$	$0.868_{\pm 0.009}$	$0.648_{\pm 0.020}$	$0.950_{\pm 0.004}$	$0.957_{\pm 0.003}$	$0.813_{\pm 0.006}$	$0.969_{\pm 0.001}$	$0.961_{\pm 0.002}$
TVAE	$0.941_{\pm 0.003}$	$0.929_{\pm 0.008}$	$0.986_{\pm 0.003}$	$0.957_{\pm 0.002}$	$0.660_{\pm 0.020}$	$0.972_{\pm 0.015}$	$0.925_{\pm 0.010}$	$0.787_{\pm 0.039}$	$0.985_{\pm 0.002}$	$0.946_{\pm 0.018}$
TVAE (org.)	$0.921_{\pm 0.007}$	$0.942_{\pm 0.003}$	$0.926_{\pm 0.008}$	$0.888_{\pm 0.014}$	$0.649_{\pm 0.019}$	$0.969_{\pm 0.003}$	$0.954_{\pm 0.005}$	$0.857_{\pm 0.015}$	$0.966_{\pm 0.004}$	$0.901_{\pm 0.008}$
ForestDiffusion	$0.947_{\pm 0.002}$	$0.817_{\pm 0.003}$	$0.972_{\pm 0.001}$	$0.740_{\pm 0.014}$	$0.661_{\pm 0.017}$	$0.966_{\pm 0.002}$	$0.881_{\pm 0.002}$	$0.804_{\pm 0.009}$	$0.987_{\pm 0.001}$	$0.987_{\pm 0.001}$
ForestFlow	$0.982_{\pm 0.002}$	$0.966_{\pm 0.001}$	$0.973_{\pm 0.002}$	$0.986_{\pm 0.002}$	$0.946_{\pm 0.009}$	$0.989_{\pm 0.002}$	$0.912_{\pm 0.001}$	$0.941_{\pm 0.013}$	$0.989_{\pm 0.001}$	$\mathbf{0.994_{\pm 0.001}}$
TabDDPM	$0.953_{\pm 0.014}$	$0.907_{\pm 0.025}$	$0.986_{\pm 0.002}$	$0.990_{\pm 0.001}$	$\mathbf{0.975_{\pm 0.011}}$	$0.986_{\pm 0.003}$	$0.992_{\pm 0.001}$	$0.971_{\pm 0.009}$	$0.973_{\pm 0.003}$	$0.983_{\pm 0.001}$
TabSyn	$0.919_{\pm 0.025}$	$0.980_{\pm 0.004}$	$0.971_{\pm 0.001}$	$0.991_{\pm 0.003}$	$0.903_{\pm 0.023}$	$0.987_{\pm 0.002}$	$0.989_{\pm 0.002}$	$0.791_{\pm 0.069}$	$0.981_{\pm 0.003}$	$0.968_{\pm 0.004}$
TabSyn (org.)	$0.958_{\pm 0.002}$	$\mathbf{0.988_{\pm 0.002}}$	$0.985_{\pm 0.002}$	$0.843_{\pm 0.214}$	$0.740_{\pm 0.174}$	$0.994_{\pm 0.001}$	$0.914_{\pm 0.110}$	$0.974_{\pm 0.005}$	$\mathbf{0.990_{\pm 0.002}}$	$0.984_{\pm 0.002}$
TabDiff	$0.911_{\pm 0.003}$	$0.962_{\pm 0.012}$	$0.975_{\pm 0.005}$	$0.988_{\pm 0.001}$	$0.651_{\pm 0.025}$	$0.977_{\pm 0.011}$	$0.990_{\pm 0.002}$	$0.776_{\pm 0.025}$	$0.974_{\pm 0.003}$	$0.988_{\pm 0.003}$
TabDiff (org.)	$0.941_{\pm 0.008}$	$0.953_{\pm 0.053}$	$\mathbf{0.989_{\pm 0.001}}$	$0.991_{\pm 0.002}$	$0.972_{\pm 0.015}$	$\mathbf{0.995_{\pm 0.001}}$	$\mathbf{0.994_{\pm 0.001}}$	$\mathbf{0.979_{\pm 0.003}}$	$0.985_{\pm 0.003}$	$0.990_{\pm 0.001}$
CDTD	$\mathbf{0.985_{\pm 0.003}}$	$0.986_{\pm 0.001}$	$0.988_{\pm 0.001}$	$\mathbf{0.993_{\pm 0.001}}$	$0.912_{\pm 0.033}$	$0.985_{\pm 0.002}$	$0.990_{\pm 0.002}$	$0.960_{\pm 0.004}$	$0.988_{\pm 0.002}$	$0.993_{\pm 0.001}$
XGB-DDPM	$0.953_{\pm 0.003}$	$0.940_{\pm 0.002}$	$0.988_{\pm 0.001}$	$0.992_{\pm 0.000}$	$0.787_{\pm 0.020}$	$0.974_{\pm 0.001}$	$\mathbf{0.994_{\pm 0.001}}$	$0.963_{\pm 0.006}$	$0.987_{\pm 0.001}$	$0.979_{\pm 0.002}$

Table 15: **MI (test)** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.881_{\pm 0.003}$	$0.882_{\pm 0.002}$	$0.943_{\pm 0.003}$	$0.968_{\pm 0.002}$	$0.928_{\pm 0.012}$	$0.952_{\pm 0.005}$	$0.952_{\pm 0.007}$	$0.915_{\pm 0.008}$	$0.887_{\pm 0.003}$	$0.973_{\pm 0.002}$
CTGAN	$0.892_{\pm 0.003}$	$0.939_{\pm 0.017}$	$0.963_{\pm 0.003}$	$0.902_{\pm 0.007}$	$0.682_{\pm 0.011}$	$0.951_{\pm 0.016}$	$0.921_{\pm 0.007}$	$0.857_{\pm 0.008}$	$0.902_{\pm 0.007}$	$0.964_{\pm 0.001}$
CTGAN (org.)	$0.813_{\pm 0.004}$	$0.935_{\pm 0.004}$	$0.936_{\pm 0.003}$	$0.867_{\pm 0.008}$	$0.679_{\pm 0.011}$	$0.948_{\pm 0.002}$	$0.957_{\pm 0.003}$	$0.799_{\pm 0.006}$	$0.881_{\pm 0.005}$	$0.958_{\pm 0.002}$
TVAE	$0.895_{\pm 0.004}$	$0.930_{\pm 0.007}$	$0.962_{\pm 0.004}$	$0.956_{\pm 0.002}$	$0.687_{\pm 0.023}$	$0.971_{\pm 0.007}$	$0.926_{\pm 0.009}$	$0.769_{\pm 0.038}$	$0.902_{\pm 0.005}$	$0.949_{\pm 0.018}$
TVAE (org.)	$0.875_{\pm 0.005}$	$0.942_{\pm 0.003}$	$0.934_{\pm 0.006}$	$0.888_{\pm 0.013}$	$0.679_{\pm 0.012}$	$0.969_{\pm 0.004}$	$0.955_{\pm 0.006}$	$0.838_{\pm 0.016}$	$0.883_{\pm 0.010}$	$0.907_{\pm 0.007}$
ForestDiffusion	$0.892_{\pm 0.003}$	$0.814_{\pm 0.006}$	$0.945_{\pm 0.002}$	$0.740_{\pm 0.015}$	$0.691_{\pm 0.011}$	$0.954_{\pm 0.004}$	$0.884_{\pm 0.003}$	$0.786_{\pm 0.011}$	$0.896_{\pm 0.003}$	$0.980_{\pm 0.002}$
ForestFlow	$0.924_{\pm 0.004}$	$0.959_{\pm 0.003}$	$0.946_{\pm 0.002}$	$0.986_{\pm 0.002}$	$0.955_{\pm 0.004}$	$0.984_{\pm 0.001}$	$0.914_{\pm 0.001}$	$0.918_{\pm 0.015}$	$0.905_{\pm 0.004}$	$0.987_{\pm 0.001}$
TabDDPM	$0.926_{\pm 0.003}$	$0.913_{\pm 0.024}$	$\mathbf{0.969_{\pm 0.002}}$	$0.989_{\pm 0.001}$	$0.963_{\pm 0.025}$	$0.987_{\pm 0.002}$	$0.989_{\pm 0.002}$	$0.961_{\pm 0.008}$	$\mathbf{0.928_{\pm 0.004}}$	$0.978_{\pm 0.002}$
TabSyn	$0.868_{\pm 0.026}$	$0.974_{\pm 0.003}$	$0.943_{\pm 0.004}$	$0.990_{\pm 0.003}$	$0.926_{\pm 0.020}$	$0.975_{\pm 0.003}$	$0.987_{\pm 0.002}$	$0.775_{\pm 0.074}$	$0.900_{\pm 0.003}$	$0.964_{\pm 0.003}$
TabSyn (org.)	$0.905_{\pm 0.005}$	$0.982_{\pm 0.001}$	$0.962_{\pm 0.004}$	$0.844_{\pm 0.213}$	$0.751_{\pm 0.156}$	$0.981_{\pm 0.003}$	$0.917_{\pm 0.103}$	$\mathbf{0.966_{\pm 0.005}}$	$0.906_{\pm 0.003}$	$0.978_{\pm 0.001}$
TabDiff	$0.860_{\pm 0.006}$	$0.956_{\pm 0.014}$	$0.947_{\pm 0.003}$	$0.986_{\pm 0.001}$	$0.681_{\pm 0.023}$	$0.965_{\pm 0.008}$	$0.988_{\pm 0.002}$	$0.760_{\pm 0.018}$	$0.899_{\pm 0.004}$	$0.982_{\pm 0.003}$
TabDiff (org.)	$0.888_{\pm 0.009}$	$0.947_{\pm 0.054}$	$0.963_{\pm 0.004}$	$0.989_{\pm 0.001}$	$\mathbf{0.975_{\pm 0.011}}$	$0.982_{\pm 0.003}$	$\mathbf{0.994_{\pm 0.001}}$	$0.964_{\pm 0.007}$	$0.904_{\pm 0.004}$	$0.983_{\pm 0.001}$
CDTD	$\mathbf{0.941_{\pm 0.005}}$	$\mathbf{0.987_{\pm 0.001}}$	$0.968_{\pm 0.002}$	$\mathbf{0.994_{\pm 0.002}}$	$0.892_{\pm 0.041}$	$\mathbf{0.988_{\pm 0.002}}$	$0.991_{\pm 0.002}$	$0.954_{\pm 0.005}$	$0.910_{\pm 0.003}$	$\mathbf{0.989_{\pm 0.002}}$
XGB-DDPM	$0.897_{\pm 0.004}$	$0.934_{\pm 0.002}$	$0.957_{\pm 0.003}$	$0.991_{\pm 0.001}$	$0.813_{\pm 0.018}$	$0.962_{\pm 0.004}$	$\mathbf{0.994_{\pm 0.001}}$	$0.944_{\pm 0.009}$	$0.898_{\pm 0.003}$	$0.973_{\pm 0.002}$

Table 16: α**-Precision** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.963_{\pm 0.014}$	$0.933_{\pm 0.005}$	$0.968_{\pm 0.014}$	$0.994_{\pm 0.003}$	$0.963_{\pm 0.004}$	$0.969_{\pm 0.007}$	$0.993_{\pm 0.003}$	$0.986_{\pm 0.004}$	$0.917_{\pm 0.018}$	$0.986_{\pm 0.003}$
CTGAN	$0.744_{\pm 0.034}$	$0.887_{\pm 0.055}$	$0.690_{\pm 0.097}$	$0.879_{\pm 0.024}$	$0.636_{\pm 0.018}$	$0.785_{\pm 0.018}$	$0.892_{\pm 0.069}$	$0.730_{\pm 0.007}$	$0.915_{\pm 0.027}$	$0.831_{\pm 0.054}$
CTGAN (org.)	$0.652_{\pm 0.047}$	$0.829_{\pm 0.017}$	$0.899_{\pm 0.050}$	$0.904_{\pm 0.009}$	$0.642_{\pm 0.022}$	$0.781_{\pm 0.027}$	$0.811_{\pm 0.062}$	$0.745_{\pm 0.009}$	$0.586_{\pm 0.037}$	$0.882_{\pm 0.053}$
TVAE	$0.920_{\pm 0.035}$	$0.772_{\pm 0.011}$	$0.897_{\pm 0.032}$	$0.948_{\pm 0.004}$	$0.800_{\pm 0.102}$	$0.979_{\pm 0.006}$	$0.665_{\pm 0.023}$	$0.808_{\pm 0.044}$	$0.903_{\pm 0.052}$	$0.473_{\pm 0.147}$
TVAE (org.)	$0.857_{\pm 0.019}$	$0.820_{\pm 0.038}$	$0.200_{\pm 0.018}$	$0.960_{\pm 0.033}$	$0.908_{\pm 0.032}$	$0.947_{\pm 0.030}$	$0.797_{\pm 0.042}$	$0.842_{\pm 0.018}$	$0.867_{\pm 0.045}$	$0.556_{\pm 0.039}$
ForestDiffusion	$0.972_{\pm 0.006}$	$0.868_{\pm 0.013}$	$0.931_{\pm 0.014}$	$0.944_{\pm 0.016}$	$0.644_{\pm 0.052}$	$\mathbf{0.985_{\pm 0.004}}$	$0.803_{\pm 0.004}$	$0.898_{\pm 0.003}$	$0.919_{\pm 0.019}$	$0.987_{\pm 0.005}$
ForestFlow	$0.961_{\pm 0.016}$	$0.951_{\pm 0.005}$	$0.933_{\pm 0.012}$	$0.965_{\pm 0.004}$	$0.933_{\pm 0.009}$	$0.915_{\pm 0.008}$	$0.860_{\pm 0.003}$	$0.957_{\pm 0.006}$	$0.926_{\pm 0.021}$	$0.966_{\pm 0.006}$
TabDDPM	$0.962_{\pm 0.011}$	$0.959_{\pm 0.024}$	$0.898_{\pm 0.029}$	$0.994_{\pm 0.001}$	$0.991_{\pm 0.003}$	$0.957_{\pm 0.010}$	$0.987_{\pm 0.004}$	$0.986_{\pm 0.006}$	$\mathbf{0.972_{\pm 0.010}}$	$0.942_{\pm 0.014}$
TabSyn	$0.932_{\pm 0.037}$	$0.955_{\pm 0.015}$	$0.545_{\pm 0.084}$	$0.994_{\pm 0.002}$	$0.982_{\pm 0.007}$	$0.984_{\pm 0.007}$	$\mathbf{0.994_{\pm 0.003}}$	$0.690_{\pm 0.100}$	$0.730_{\pm 0.118}$	$0.902_{\pm 0.054}$
TabSyn (org.)	$\mathbf{0.979_{\pm 0.007}}$	$\mathbf{0.977_{\pm 0.010}}$	$0.811_{\pm 0.017}$	$0.841_{\pm 0.224}$	$0.958_{\pm 0.030}$	$\mathbf{0.985_{\pm 0.006}}$	$0.875_{\pm 0.170}$	$0.990_{\pm 0.002}$	$0.969_{\pm 0.012}$	$\mathbf{0.988_{\pm 0.003}}$
TabDiff	$0.968_{\pm 0.009}$	$0.941_{\pm 0.012}$	$0.873_{\pm 0.018}$	$\mathbf{0.995_{\pm 0.002}}$	$0.842_{\pm 0.076}$	$0.949_{\pm 0.036}$	$0.981_{\pm 0.022}$	$0.582_{\pm 0.024}$	$0.504_{\pm 0.020}$	$0.832_{\pm 0.009}$
TabDiff (org.)	$0.975_{\pm 0.006}$	$0.881_{\pm 0.125}$	$0.956_{\pm 0.020}$	$\mathbf{0.995_{\pm 0.002}}$	$\mathbf{0.996_{\pm 0.001}}$	$0.980_{\pm 0.006}$	$0.990_{\pm 0.003}$	$\mathbf{0.991_{\pm 0.003}}$	$0.836_{\pm 0.090}$	$0.940_{\pm 0.074}$
CDTD	$\mathbf{0.979_{\pm 0.012}}$	$0.960_{\pm 0.003}$	$\mathbf{0.971_{\pm 0.016}}$	$0.992_{\pm 0.002}$	$0.967_{\pm 0.004}$	$0.952_{\pm 0.011}$	$0.992_{\pm 0.002}$	$0.968_{\pm 0.006}$	$0.941_{\pm 0.025}$	$0.942_{\pm 0.006}$
XGB-DDPM	$0.952_{\pm 0.015}$	$0.948_{\pm 0.003}$	$0.880_{\pm 0.018}$	$0.960_{\pm 0.003}$	$0.965_{\pm 0.004}$	$0.899_{\pm 0.007}$	$0.979_{\pm 0.003}$	$0.959_{\pm 0.007}$	$0.965_{\pm 0.015}$	$0.972_{\pm 0.006}$

Table 17: β**-Recall** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.747_{\pm 0.034}$	$0.672_{\pm 0.007}$	$0.937_{\pm 0.012}$	$0.893_{\pm 0.004}$	$0.955_{\pm 0.002}$	$0.817_{\pm 0.019}$	$0.958_{\pm 0.002}$	$0.637_{\pm 0.024}$	$0.889_{\pm 0.020}$	$0.972_{\pm 0.002}$
CTGAN	$0.663_{\pm 0.020}$	$0.617_{\pm 0.015}$	$0.786_{\pm 0.055}$	$0.420_{\pm 0.012}$	$0.662_{\pm 0.030}$	$0.452_{\pm 0.047}$	$0.701_{\pm 0.005}$	$0.088_{\pm 0.013}$	$0.832_{\pm 0.046}$	$0.812_{\pm 0.016}$
CTGAN (org.)	$0.263_{\pm 0.037}$	$0.562_{\pm 0.011}$	$0.674_{\pm 0.022}$	$0.413_{\pm 0.025}$	$0.801_{\pm 0.017}$	$0.314_{\pm 0.056}$	$0.840_{\pm 0.014}$	$0.018_{\pm 0.010}$	$0.405_{\pm 0.043}$	$0.811_{\pm 0.020}$
TVAE	$0.659_{\pm 0.018}$	$0.335_{\pm 0.028}$	$0.962_{\pm 0.024}$	$0.442_{\pm 0.021}$	$0.359_{\pm 0.040}$	$0.666_{\pm 0.129}$	$0.151_{\pm 0.031}$	$0.100_{\pm 0.040}$	$0.952_{\pm 0.013}$	$0.127_{\pm 0.094}$
TVAE (org.)	$0.503_{\pm 0.022}$	$0.457_{\pm 0.040}$	$0.730_{\pm 0.020}$	$0.431_{\pm 0.069}$	$0.713_{\pm 0.026}$	$0.678_{\pm 0.026}$	$0.600_{\pm 0.028}$	$0.083_{\pm 0.013}$	$0.816_{\pm 0.035}$	$0.217_{\pm 0.014}$
ForestDiffusion	$0.719_{\pm 0.023}$	$0.206_{\pm 0.012}$	$0.759_{\pm 0.022}$	$0.244_{\pm 0.013}$	$0.284_{\pm 0.010}$	$0.736_{\pm 0.014}$	$0.265_{\pm 0.004}$	$0.044_{\pm 0.005}$	$0.989_{\pm 0.005}$	$0.969_{\pm 0.003}$
ForestFlow	$0.896_{\pm 0.016}$	$0.947_{\pm 0.002}$	$0.778_{\pm 0.017}$	$0.869_{\pm 0.003}$	$0.774_{\pm 0.054}$	$0.912_{\pm 0.004}$	$0.311_{\pm 0.001}$	$0.755_{\pm 0.028}$	$0.994_{\pm 0.003}$	$0.973_{\pm 0.003}$
TabDDPM	$0.863_{\pm 0.016}$	$0.907_{\pm 0.013}$	$\mathbf{0.996_{\pm 0.002}}$	$0.896_{\pm 0.004}$	$\mathbf{0.968_{\pm 0.001}}$	$0.907_{\pm 0.015}$	$0.972_{\pm 0.002}$	$\mathbf{0.905_{\pm 0.004}}$	$\mathbf{0.996_{\pm 0.003}}$	$0.969_{\pm 0.004}$
TabSyn	$0.690_{\pm 0.058}$	$0.925_{\pm 0.004}$	$0.778_{\pm 0.036}$	$0.914_{\pm 0.016}$	$0.967_{\pm 0.002}$	$0.803_{\pm 0.011}$	$0.961_{\pm 0.003}$	$0.149_{\pm 0.181}$	$0.755_{\pm 0.059}$	$0.917_{\pm 0.032}$
TabSyn (org.)	$0.794_{\pm 0.021}$	$0.950_{\pm 0.004}$	$0.992_{\pm 0.004}$	$0.628_{\pm 0.431}$	$0.791_{\pm 0.127}$	$0.855_{\pm 0.009}$	$0.792_{\pm 0.249}$	$0.848_{\pm 0.007}$	$0.980_{\pm 0.007}$	$0.970_{\pm 0.002}$
TabDiff	$0.617_{\pm 0.021}$	$0.859_{\pm 0.064}$	$0.861_{\pm 0.057}$	$0.929_{\pm 0.003}$	$0.627_{\pm 0.039}$	$0.753_{\pm 0.064}$	$0.946_{\pm 0.035}$	$0.040_{\pm 0.005}$	$0.460_{\pm 0.038}$	$0.898_{\pm 0.006}$
TabDiff (org.)	$0.774_{\pm 0.024}$	$0.779_{\pm 0.271}$	$0.991_{\pm 0.004}$	$0.943_{\pm 0.006}$	$0.966_{\pm 0.002}$	$0.901_{\pm 0.007}$	$\mathbf{0.978_{\pm 0.002}}$	$0.870_{\pm 0.007}$	$0.865_{\pm 0.072}$	$0.948_{\pm 0.034}$
CDTD	$\mathbf{0.981_{\pm 0.007}}$	$\mathbf{0.972_{\pm 0.002}}$	$0.995_{\pm 0.002}$	$\mathbf{0.962_{\pm 0.002}}$	$\mathbf{0.968_{\pm 0.002}}$	$\mathbf{0.994_{\pm 0.002}}$	$0.974_{\pm 0.002}$	$0.877_{\pm 0.005}$	$0.995_{\pm 0.002}$	$\mathbf{0.991_{\pm 0.001}}$
XGB-DDPM	$0.827_{\pm 0.019}$	$0.898_{\pm 0.003}$	$0.988_{\pm 0.004}$	$0.922_{\pm 0.002}$	$0.966_{\pm 0.002}$	$0.853_{\pm 0.007}$	$0.966_{\pm 0.002}$	$0.863_{\pm 0.007}$	$0.984_{\pm 0.006}$	$0.968_{\pm 0.003}$

Table 18: **MLE** scores for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500 R^2	default AUC	credit-g AUC	diamonds R^2	give-credit AUC	wine R^2	bank AUC	jm1 AUC	diabetes AUC	shipping AUC
Real	$0.843_{\pm 0.007}$	$0.776_{\pm 0.005}$	$0.739_{\pm 0.023}$	$0.981_{\pm 0.001}$	$0.859_{\pm 0.006}$	$0.451_{\pm 0.024}$	$0.769_{\pm 0.002}$	$0.730_{\pm 0.006}$	$0.834_{\pm 0.016}$	$0.739_{\pm 0.008}$
ARF	$0.813_{\pm 0.018}$	$0.764_{\pm 0.005}$	$0.649_{\pm 0.048}$	$0.965_{\pm 0.001}$	$0.854_{\pm 0.006}$	$0.285_{\pm 0.014}$	$0.761_{\pm 0.003}$	$\mathbf{0.712_{\pm 0.007}}$	$0.785_{\pm 0.034}$	$0.737_{\pm 0.005}$
CTGAN	$0.779_{\pm 0.016}$	$0.727_{\pm 0.016}$	$0.688_{\pm 0.031}$	$0.897_{\pm 0.006}$	$0.810_{\pm 0.011}$	$0.114_{\pm 0.046}$	$0.559_{\pm 0.038}$	$0.690_{\pm 0.018}$	$0.775_{\pm 0.030}$	$0.717_{\pm 0.012}$
CTGAN (org.)	$0.032_{\pm 0.074}$	$0.728_{\pm 0.009}$	$0.493_{\pm 0.056}$	$0.875_{\pm 0.010}$	$0.843_{\pm 0.006}$	$-0.097_{\pm 0.088}$	$0.736_{\pm 0.006}$	$0.676_{\pm 0.020}$	$0.555_{\pm 0.077}$	$0.725_{\pm 0.009}$
TVAE	$0.771_{\pm 0.019}$	$0.739_{\pm 0.006}$	$0.723_{\pm 0.024}$	$0.895_{\pm 0.012}$	$0.648_{\pm 0.112}$	$0.268_{\pm 0.046}$	$0.622_{\pm 0.080}$	$0.682_{\pm 0.015}$	$0.765_{\pm 0.040}$	$0.638_{\pm 0.100}$
TVAE (org.)	$0.771_{\pm 0.027}$	$0.750_{\pm 0.005}$	$0.684_{\pm 0.023}$	$0.899_{\pm 0.014}$	$0.791_{\pm 0.039}$	$0.262_{\pm 0.023}$	$0.724_{\pm 0.005}$	$0.665_{\pm 0.021}$	$0.804_{\pm 0.028}$	$0.736_{\pm 0.008}$
ForestDiffusion	$0.823_{\pm 0.011}$	$0.676_{\pm 0.012}$	$0.723_{\pm 0.028}$	$0.964_{\pm 0.002}$	$0.679_{\pm 0.022}$	$0.304_{\pm 0.016}$	$0.742_{\pm 0.006}$	$0.645_{\pm 0.020}$	$0.819_{\pm 0.022}$	$0.736_{\pm 0.008}$
ForestFlow	$0.825_{\pm 0.010}$	$0.708_{\pm 0.007}$	$0.719_{\pm 0.032}$	$0.970_{\pm 0.001}$	$0.729_{\pm 0.017}$	$0.312_{\pm 0.025}$	$0.748_{\pm 0.006}$	$0.699_{\pm 0.010}$	$0.825_{\pm 0.020}$	$0.737_{\pm 0.008}$
TabDDPM	$\mathbf{0.831_{\pm 0.009}}$	$0.695_{\pm 0.039}$	$\mathbf{0.733_{\pm 0.022}}$	$0.974_{\pm 0.002}$	$\mathbf{0.858_{\pm 0.005}}$	$0.323_{\pm 0.018}$	$0.763_{\pm 0.004}$	$0.706_{\pm 0.009}$	$\mathbf{0.826_{\pm 0.021}}$	$0.735_{\pm 0.006}$
TabSyn	$0.794_{\pm 0.047}$	$0.767_{\pm 0.009}$	$0.661_{\pm 0.048}$	$0.975_{\pm 0.003}$	$0.856_{\pm 0.003}$	$0.295_{\pm 0.024}$	$0.756_{\pm 0.006}$	$0.703_{\pm 0.006}$	$0.802_{\pm 0.025}$	$0.734_{\pm 0.009}$
TabSyn (org.)	$0.826_{\pm 0.014}$	$0.769_{\pm 0.007}$	$0.726_{\pm 0.034}$	$0.647_{\pm 0.476}$	$0.614_{\pm 0.187}$	$0.316_{\pm 0.018}$	$0.679_{\pm 0.122}$	$0.700_{\pm 0.010}$	$0.818_{\pm 0.017}$	$0.741_{\pm 0.010}$
TabDiff	$0.804_{\pm 0.013}$	$\mathbf{0.771_{\pm 0.004}}$	$0.575_{\pm 0.104}$	$0.976_{\pm 0.002}$	$0.801_{\pm 0.063}$	$0.215_{\pm 0.066}$	$0.763_{\pm 0.003}$	$0.702_{\pm 0.007}$	$0.802_{\pm 0.024}$	$\mathbf{0.743_{\pm 0.008}}$
TabDiff (org.)	$0.814_{\pm 0.012}$	$0.765_{\pm 0.004}$	$0.717_{\pm 0.027}$	$0.977_{\pm 0.001}$	$0.856_{\pm 0.005}$	$0.316_{\pm 0.016}$	$0.762_{\pm 0.004}$	$0.709_{\pm 0.007}$	$0.813_{\pm 0.021}$	$0.740_{\pm 0.007}$
CDTD	$0.830_{\pm 0.013}$	$\mathbf{0.771_{\pm 0.008}}$	$0.732_{\pm 0.024}$	$\mathbf{0.978_{\pm 0.002}}$	$\mathbf{0.858_{\pm 0.005}}$	$\mathbf{0.341_{\pm 0.025}}$	$0.760_{\pm 0.004}$	$0.702_{\pm 0.011}$	$0.820_{\pm 0.020}$	$0.738_{\pm 0.007}$
XGB-DDPM	$0.810_{\pm 0.012}$	$0.764_{\pm 0.007}$	$0.726_{\pm 0.031}$	$0.973_{\pm 0.002}$	$0.857_{\pm 0.005}$	$0.292_{\pm 0.015}$	$\mathbf{0.765_{\pm 0.003}}$	$0.707_{\pm 0.008}$	$0.814_{\pm 0.018}$	$0.738_{\pm 0.007}$

Table 19: **DCR Share** for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined. Any DCR Share < 0.5 is similarly considered as no sign of overfitting.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.536_{\pm 0.019}$	$0.514_{\pm 0.004}$	$0.571_{\pm 0.020}$	$0.535_{\pm 0.003}$	$0.504_{\pm 0.004}$	$0.536_{\pm 0.008}$	$0.528_{\pm 0.004}$	$0.532_{\pm 0.007}$	$0.529_{\pm 0.021}$	$0.523_{\pm 0.007}$
CTGAN	$0.520_{\pm 0.021}$	$0.509_{\pm 0.008}$	$0.528_{\pm 0.035}$	$0.511_{\pm 0.006}$	$0.503_{\pm 0.005}$	$\mathbf{0.507_{\pm 0.011}}$	$0.502_{\pm 0.010}$	$0.502_{\pm 0.005}$	$0.544_{\pm 0.033}$	$0.515_{\pm 0.011}$
CTGAN (org.)	$\mathbf{0.471_{\pm 0.023}}$	$0.507_{\pm 0.006}$	$0.524_{\pm 0.029}$	$\mathbf{0.504_{\pm 0.005}}$	$0.507_{\pm 0.004}$	$\mathbf{0.507_{\pm 0.021}}$	$\mathbf{0.501_{\pm 0.003}}$	$0.505_{\pm 0.018}$	$0.521_{\pm 0.047}$	$\mathbf{0.501_{\pm 0.007}}$
TVAE	$0.538_{\pm 0.021}$	$0.529_{\pm 0.025}$	$0.642_{\pm 0.020}$	$0.510_{\pm 0.005}$	$0.502_{\pm 0.006}$	$0.523_{\pm 0.029}$	$0.516_{\pm 0.029}$	$0.515_{\pm 0.008}$	$0.618_{\pm 0.039}$	$0.667_{\pm 0.201}$
TVAE (org.)	$\mathbf{0.478_{\pm 0.021}}$	$0.515_{\pm 0.017}$	$0.532_{\pm 0.102}$	$0.512_{\pm 0.008}$	$0.501_{\pm 0.004}$	$0.520_{\pm 0.019}$	$0.509_{\pm 0.006}$	$\mathbf{0.501_{\pm 0.009}}$	$\mathbf{0.494_{\pm 0.043}}$	$0.527_{\pm 0.024}$
ForestDiffusion	$0.529_{\pm 0.019}$	$0.509_{\pm 0.004}$	$0.545_{\pm 0.024}$	$0.506_{\pm 0.003}$	$0.501_{\pm 0.006}$	$0.523_{\pm 0.010}$	$0.507_{\pm 0.007}$	$0.515_{\pm 0.008}$	$0.585_{\pm 0.019}$	$0.514_{\pm 0.005}$
ForestFlow	$0.542_{\pm 0.016}$	$0.510_{\pm 0.004}$	$0.544_{\pm 0.025}$	$0.507_{\pm 0.004}$	$0.503_{\pm 0.003}$	$0.532_{\pm 0.012}$	$0.509_{\pm 0.009}$	$0.515_{\pm 0.007}$	$0.602_{\pm 0.019}$	$0.510_{\pm 0.006}$
TabDDPM	$0.550_{\pm 0.020}$	$\mathbf{0.504_{\pm 0.013}}$	$0.641_{\pm 0.020}$	$0.506_{\pm 0.003}$	$\mathbf{0.500_{\pm 0.003}}$	$0.534_{\pm 0.015}$	$0.513_{\pm 0.003}$	$0.527_{\pm 0.007}$	$0.627_{\pm 0.023}$	$0.513_{\pm 0.007}$
TabSyn	$0.501_{\pm 0.020}$	$0.505_{\pm 0.005}$	$0.566_{\pm 0.035}$	$0.506_{\pm 0.004}$	$0.501_{\pm 0.004}$	$0.508_{\pm 0.013}$	$0.505_{\pm 0.003}$	$0.514_{\pm 0.008}$	$0.539_{\pm 0.038}$	$0.510_{\pm 0.007}$
TabSyn (org.)	$0.520_{\pm 0.019}$	$0.505_{\pm 0.004}$	$0.617_{\pm 0.038}$	$0.510_{\pm 0.005}$	$0.505_{\pm 0.013}$	$0.520_{\pm 0.012}$	$0.504_{\pm 0.004}$	$\mathbf{0.501_{\pm 0.006}}$	$0.547_{\pm 0.028}$	$0.510_{\pm 0.006}$
TabDiff	$\mathbf{0.485_{\pm 0.017}}$	$0.517_{\pm 0.003}$	$\mathbf{0.512_{\pm 0.028}}$	$0.511_{\pm 0.004}$	$0.503_{\pm 0.015}$	$0.513_{\pm 0.007}$	$0.511_{\pm 0.004}$	$0.507_{\pm 0.009}$	$0.542_{\pm 0.041}$	$0.508_{\pm 0.007}$
TabDiff (org.)	$\mathbf{0.500_{\pm 0.019}}$	$0.519_{\pm 0.012}$	$0.629_{\pm 0.032}$	$0.519_{\pm 0.005}$	$0.501_{\pm 0.003}$	$0.533_{\pm 0.008}$	$0.532_{\pm 0.005}$	$0.505_{\pm 0.006}$	$0.521_{\pm 0.034}$	$0.510_{\pm 0.007}$
CDTD	$0.612_{\pm 0.019}$	$0.520_{\pm 0.005}$	$0.636_{\pm 0.028}$	$0.525_{\pm 0.004}$	$0.501_{\pm 0.003}$	$0.622_{\pm 0.008}$	$0.529_{\pm 0.004}$	$0.505_{\pm 0.008}$	$0.628_{\pm 0.028}$	$0.590_{\pm 0.006}$
XGB-DDPM	$0.513_{\pm 0.017}$	$0.506_{\pm 0.006}$	$0.587_{\pm 0.028}$	$0.505_{\pm 0.003}$	$0.501_{\pm 0.003}$	$0.518_{\pm 0.012}$	$0.506_{\pm 0.004}$	$0.511_{\pm 0.006}$	$0.556_{\pm 0.025}$	$0.512_{\pm 0.008}$

Table 20: **MIA** AUC for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined. Any AUC < 0.5 is similarly considered as no increased membership disclosure risk.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.514_{\pm 0.021}$	$0.501_{\pm 0.007}$	$0.513_{\pm 0.017}$	$0.511_{\pm 0.003}$	$0.504_{\pm 0.002}$	$0.509_{\pm 0.010}$	$0.508_{\pm 0.004}$	$0.507_{\pm 0.007}$	$0.516_{\pm 0.041}$	$0.513_{\pm 0.006}$
CTGAN	$0.503_{\pm 0.012}$	$\mathbf{0.499_{\pm 0.007}}$	$\mathbf{0.480_{\pm 0.018}}$	$\mathbf{0.502_{\pm 0.004}}$	$\mathbf{0.500_{\pm 0.003}}$	$0.507_{\pm 0.006}$	$0.502_{\pm 0.003}$	$0.503_{\pm 0.012}$	$0.534_{\pm 0.020}$	$0.505_{\pm 0.007}$
CTGAN (org.)	$0.515_{\pm 0.014}$	$\mathbf{0.498_{\pm 0.006}}$	$0.516_{\pm 0.014}$	$0.504_{\pm 0.002}$	$\mathbf{0.500_{\pm 0.002}}$	$\mathbf{0.505_{\pm 0.039}}$	$0.505_{\pm 0.005}$	$0.506_{\pm 0.006}$	$0.515_{\pm 0.019}$	$\mathbf{0.501_{\pm 0.005}}$
TVAE	$0.501_{\pm 0.020}$	$0.503_{\pm 0.003}$	$0.529_{\pm 0.021}$	$0.504_{\pm 0.002}$	$\mathbf{0.500_{\pm 0.002}}$	$0.510_{\pm 0.015}$	$\mathbf{0.500_{\pm 0.003}}$	$0.501_{\pm 0.012}$	$0.537_{\pm 0.025}$	$0.503_{\pm 0.005}$
TVAE (org.)	$\mathbf{0.489_{\pm 0.017}}$	$0.503_{\pm 0.004}$	$0.508_{\pm 0.017}$	$0.504_{\pm 0.002}$	$\mathbf{0.500_{\pm 0.002}}$	$0.518_{\pm 0.007}$	$\mathbf{0.500_{\pm 0.005}}$	$\mathbf{0.500_{\pm 0.012}}$	$\mathbf{0.491_{\pm 0.023}}$	$0.510_{\pm 0.008}$
ForestDiffusion	$0.513_{\pm 0.020}$	$\mathbf{0.499_{\pm 0.006}}$	$0.501_{\pm 0.020}$	$\mathbf{0.502_{\pm 0.003}}$	$\mathbf{0.500_{\pm 0.002}}$	$\mathbf{0.505_{\pm 0.011}}$	$0.503_{\pm 0.004}$	$\mathbf{0.500_{\pm 0.008}}$	$0.521_{\pm 0.026}$	$0.514_{\pm 0.005}$
ForestFlow	$0.505_{\pm 0.012}$	$0.501_{\pm 0.004}$	$\mathbf{0.500_{\pm 0.022}}$	$0.505_{\pm 0.003}$	$\mathbf{0.500_{\pm 0.002}}$	$0.512_{\pm 0.007}$	$0.503_{\pm 0.003}$	$0.504_{\pm 0.009}$	$0.542_{\pm 0.015}$	$0.519_{\pm 0.005}$
TabDDPM	$0.513_{\pm 0.018}$	$0.506_{\pm 0.006}$	$0.550_{\pm 0.018}$	$0.507_{\pm 0.002}$	$0.504_{\pm 0.003}$	$0.518_{\pm 0.011}$	$0.509_{\pm 0.004}$	$0.522_{\pm 0.006}$	$0.565_{\pm 0.022}$	$0.517_{\pm 0.007}$
TabSyn	$0.501_{\pm 0.012}$	$0.507_{\pm 0.007}$	$\mathbf{0.468_{\pm 0.016}}$	$0.505_{\pm 0.003}$	$0.502_{\pm 0.002}$	$0.513_{\pm 0.011}$	$0.506_{\pm 0.004}$	$0.504_{\pm 0.007}$	$0.539_{\pm 0.027}$	$0.510_{\pm 0.006}$
TabSyn (org.)	$0.507_{\pm 0.013}$	$0.509_{\pm 0.005}$	$0.522_{\pm 0.028}$	$0.508_{\pm 0.003}$	$0.503_{\pm 0.003}$	$0.518_{\pm 0.010}$	$0.507_{\pm 0.004}$	$0.513_{\pm 0.005}$	$0.541_{\pm 0.022}$	$0.514_{\pm 0.006}$
TabDiff	$0.504_{\pm 0.013}$	$0.503_{\pm 0.004}$	$\mathbf{0.482_{\pm 0.016}}$	$0.508_{\pm 0.003}$	$\mathbf{0.500_{\pm 0.001}}$	$0.508_{\pm 0.009}$	$0.507_{\pm 0.003}$	$0.503_{\pm 0.005}$	$0.540_{\pm 0.017}$	$0.509_{\pm 0.005}$
TabDiff (org.)	$0.509_{\pm 0.013}$	$0.509_{\pm 0.005}$	$0.513_{\pm 0.011}$	$0.509_{\pm 0.002}$	$0.504_{\pm 0.002}$	$0.520_{\pm 0.007}$	$0.516_{\pm 0.003}$	$0.513_{\pm 0.006}$	$0.540_{\pm 0.021}$	$0.515_{\pm 0.007}$
CDTD	$0.523_{\pm 0.013}$	$0.516_{\pm 0.004}$	$0.545_{\pm 0.015}$	$0.514_{\pm 0.002}$	$0.503_{\pm 0.002}$	$0.540_{\pm 0.008}$	$0.511_{\pm 0.003}$	$0.514_{\pm 0.005}$	$0.553_{\pm 0.021}$	$0.535_{\pm 0.006}$
XGB-DDPM	$0.510_{\pm 0.013}$	$0.507_{\pm 0.003}$	$0.526_{\pm 0.018}$	$0.510_{\pm 0.002}$	$0.503_{\pm 0.002}$	$0.517_{\pm 0.008}$	$0.507_{\pm 0.004}$	$0.517_{\pm 0.006}$	$0.532_{\pm 0.016}$	$0.518_{\pm 0.007}$

Table 21: **MIA** Lift@10% for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined. Any Lift < 1.0 is similarly considered as no increased membership disclosure risk.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$1.015_{\pm 0.117}$	$1.016_{\pm 0.025}$	$1.041_{\pm 0.106}$	$1.040_{\pm 0.016}$	$1.014_{\pm 0.012}$	$1.059_{\pm 0.045}$	$1.034_{\pm 0.019}$	$1.027_{\pm 0.044}$	$1.061_{\pm 0.159}$	$1.056_{\pm 0.036}$
CTGAN	$1.031_{\pm 0.097}$	$1.010_{\pm 0.043}$	$\mathbf{0.939_{\pm 0.067}}$	$1.025_{\pm 0.016}$	$\mathbf{0.999_{\pm 0.011}}$	$1.010_{\pm 0.026}$	$\mathbf{0.997_{\pm 0.014}}$	$1.007_{\pm 0.030}$	$1.041_{\pm 0.162}$	$1.048_{\pm 0.049}$
CTGAN (org.)	$1.020_{\pm 0.107}$	$1.010_{\pm 0.040}$	$1.019_{\pm 0.080}$	$1.021_{\pm 0.011}$	$\mathbf{0.999_{\pm 0.012}}$	$\mathbf{1.008_{\pm 0.039}}$	$1.005_{\pm 0.017}$	$\mathbf{0.989_{\pm 0.029}}$	$1.058_{\pm 0.119}$	$\mathbf{1.006_{\pm 0.033}}$
TVAE	$1.021_{\pm 0.117}$	$1.031_{\pm 0.011}$	$1.190_{\pm 0.107}$	$\mathbf{1.005_{\pm 0.010}}$	$\mathbf{0.999_{\pm 0.010}}$	$1.050_{\pm 0.058}$	$\mathbf{0.998_{\pm 0.017}}$	$1.019_{\pm 0.030}$	$1.143_{\pm 0.132}$	$1.025_{\pm 0.025}$
TVAE (org.)	$\mathbf{0.986_{\pm 0.090}}$	$1.018_{\pm 0.026}$	$1.037_{\pm 0.150}$	$1.014_{\pm 0.012}$	$1.001_{\pm 0.010}$	$1.066_{\pm 0.047}$	$1.004_{\pm 0.018}$	$1.002_{\pm 0.033}$	$\mathbf{0.956_{\pm 0.101}}$	$1.018_{\pm 0.031}$
ForestDiffusion	$1.056_{\pm 0.099}$	$\mathbf{0.999_{\pm 0.026}}$	$\mathbf{0.991_{\pm 0.141}}$	$1.008_{\pm 0.017}$	$\mathbf{0.992_{\pm 0.009}}$	$1.022_{\pm 0.029}$	$1.003_{\pm 0.016}$	$\mathbf{0.983_{\pm 0.044}}$	$1.028_{\pm 0.118}$	$1.045_{\pm 0.031}$
ForestFlow	$1.018_{\pm 0.070}$	$1.011_{\pm 0.020}$	$\mathbf{0.988_{\pm 0.145}}$	$1.015_{\pm 0.020}$	$\mathbf{0.996_{\pm 0.011}}$	$1.060_{\pm 0.036}$	$1.009_{\pm 0.018}$	$\mathbf{0.983_{\pm 0.024}}$	$1.105_{\pm 0.116}$	$1.064_{\pm 0.028}$
TabDDPM	$1.029_{\pm 0.075}$	$1.017_{\pm 0.020}$	$1.385_{\pm 0.084}$	$1.022_{\pm 0.014}$	$1.014_{\pm 0.009}$	$1.085_{\pm 0.039}$	$1.031_{\pm 0.020}$	$1.070_{\pm 0.038}$	$1.334_{\pm 0.092}$	$1.069_{\pm 0.032}$
TabSyn	$1.017_{\pm 0.074}$	$1.020_{\pm 0.024}$	$\mathbf{0.923_{\pm 0.075}}$	$1.021_{\pm 0.011}$	$1.009_{\pm 0.010}$	$1.059_{\pm 0.043}$	$1.018_{\pm 0.020}$	$\mathbf{0.997_{\pm 0.044}}$	$1.146_{\pm 0.144}$	$1.043_{\pm 0.032}$
TabSyn (org.)	$1.045_{\pm 0.068}$	$1.031_{\pm 0.021}$	$1.064_{\pm 0.103}$	$1.032_{\pm 0.012}$	$1.016_{\pm 0.011}$	$1.078_{\pm 0.043}$	$1.030_{\pm 0.017}$	$1.044_{\pm 0.036}$	$1.103_{\pm 0.108}$	$1.037_{\pm 0.030}$
TabDiff	$\mathbf{1.000_{\pm 0.073}}$	$1.022_{\pm 0.015}$	$\mathbf{0.935_{\pm 0.074}}$	$1.029_{\pm 0.014}$	$1.004_{\pm 0.010}$	$1.044_{\pm 0.041}$	$1.028_{\pm 0.017}$	$\mathbf{0.972_{\pm 0.033}}$	$1.149_{\pm 0.105}$	$1.029_{\pm 0.033}$
TabDiff (org.)	$1.002_{\pm 0.072}$	$1.032_{\pm 0.024}$	$1.041_{\pm 0.074}$	$1.039_{\pm 0.013}$	$1.016_{\pm 0.010}$	$1.090_{\pm 0.043}$	$1.060_{\pm 0.019}$	$1.046_{\pm 0.027}$	$1.114_{\pm 0.127}$	$1.058_{\pm 0.035}$
CDTD	$1.146_{\pm 0.072}$	$1.053_{\pm 0.017}$	$1.359_{\pm 0.085}$	$1.048_{\pm 0.012}$	$1.011_{\pm 0.009}$	$1.231_{\pm 0.035}$	$1.043_{\pm 0.016}$	$1.024_{\pm 0.026}$	$1.193_{\pm 0.101}$	$1.145_{\pm 0.027}$
XGB-DDPM	$1.025_{\pm 0.054}$	$1.030_{\pm 0.015}$	$1.085_{\pm 0.107}$	$1.035_{\pm 0.013}$	$1.013_{\pm 0.009}$	$1.056_{\pm 0.042}$	$1.024_{\pm 0.017}$	$1.059_{\pm 0.026}$	$1.068_{\pm 0.107}$	$1.057_{\pm 0.027}$

Table 22: **Training Time (minutes))** for each dataset and generator. Shows average ± standard deviation over 3 training seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$\mathbf{0.013_{\pm 0.001}}$	$\mathbf{0.522_{\pm 0.037}}$	$\mathbf{0.016_{\pm 0.002}}$	$\mathbf{0.866_{\pm 0.144}}$	$\mathbf{1.007_{\pm 0.020}}$	$\mathbf{0.067_{\pm 0.001}}$	$\mathbf{0.208_{\pm 0.005}}$	$\mathbf{0.180_{\pm 0.032}}$	$\mathbf{0.011_{\pm 0.000}}$	$\mathbf{0.054_{\pm 0.000}}$
CTGAN	$13.019_{\pm 0.006}$	$17.614_{\pm 0.069}$	$15.338_{\pm 0.059}$	$14.948_{\pm 0.105}$	$14.965_{\pm 0.528}$	$14.533_{\pm 0.436}$	$15.148_{\pm 0.723}$	$17.097_{\pm 0.191}$	$14.004_{\pm 0.198}$	$14.473_{\pm 0.285}$
CTGAN (org.)	$0.119_{\pm 0.008}$	$4.387_{\pm 0.343}$	$0.313_{\pm 0.402}$	$5.615_{\pm 0.219}$	$14.815_{\pm 1.557}$	$0.838_{\pm 0.197}$	$4.683_{\pm 0.345}$	$1.295_{\pm 0.008}$	$0.260_{\pm 0.342}$	$0.898_{\pm 0.002}$
TVAE	$4.313_{\pm 0.316}$	$11.332_{\pm 0.225}$	$4.503_{\pm 0.054}$	$9.275_{\pm 0.114}$	$9.354_{\pm 0.058}$	$7.320_{\pm 0.348}$	$12.591_{\pm 0.849}$	$10.797_{\pm 0.929}$	$3.680_{\pm 0.356}$	$7.632_{\pm 0.119}$
TVAE (org.)	$0.225_{\pm 0.261}$	$2.958_{\pm 0.637}$	$0.273_{\pm 0.369}$	$2.709_{\pm 0.067}$	$7.238_{\pm 0.052}$	$0.341_{\pm 0.001}$	$2.138_{\pm 0.049}$	$0.950_{\pm 0.205}$	$0.056_{\pm 0.000}$	$0.428_{\pm 0.000}$
ForestDiffusion	$0.648_{\pm 0.881}$	$40.610_{\pm 1.398}$	$6.187_{\pm 0.904}$	$43.037_{\pm 2.120}$	$39.852_{\pm 0.341}$	$2.834_{\pm 0.834}$	$34.758_{\pm 3.032}$	$5.354_{\pm 0.669}$	$0.082_{\pm 0.001}$	$2.432_{\pm 0.059}$
ForestFlow	$0.738_{\pm 1.005}$	$20.702_{\pm 1.143}$	$4.213_{\pm 1.040}$	$32.042_{\pm 1.084}$	$20.737_{\pm 0.049}$	$1.605_{\pm 0.993}$	$30.725_{\pm 0.874}$	$6.352_{\pm 0.190}$	$0.100_{\pm 0.001}$	$2.326_{\pm 0.447}$
TabDDPM	$2.572_{\pm 0.008}$	$4.121_{\pm 0.011}$	$4.751_{\pm 0.313}$	$3.925_{\pm 0.012}$	$2.498_{\pm 0.057}$	$3.154_{\pm 0.006}$	$4.894_{\pm 0.057}$	$2.003_{\pm 0.007}$	$1.178_{\pm 0.003}$	$3.806_{\pm 0.017}$
TabSyn	$2.565_{\pm 0.021}$	$6.304_{\pm 0.034}$	$3.836_{\pm 0.100}$	$3.964_{\pm 0.010}$	$3.918_{\pm 0.054}$	$3.376_{\pm 0.009}$	$5.072_{\pm 0.001}$	$6.096_{\pm 0.378}$	$2.462_{\pm 0.115}$	$5.534_{\pm 0.040}$
TabSyn (org.)	$1.805_{\pm 0.246}$	$20.739_{\pm 0.213}$	$2.482_{\pm 0.457}$	$28.027_{\pm 0.032}$	$77.991_{\pm 1.097}$	$4.494_{\pm 0.012}$	$27.480_{\pm 0.083}$	$7.637_{\pm 0.096}$	$1.477_{\pm 0.079}$	$6.490_{\pm 0.060}$
TabDiff	$3.136_{\pm 0.001}$	$5.130_{\pm 0.009}$	$5.548_{\pm 0.029}$	$5.022_{\pm 0.035}$	$4.568_{\pm 0.019}$	$3.843_{\pm 0.004}$	$5.518_{\pm 0.699}$	$4.256_{\pm 0.030}$	$3.191_{\pm 0.014}$	$4.608_{\pm 0.004}$
TabDiff (org.)	$1.917_{\pm 0.006}$	$24.258_{\pm 0.064}$	$2.972_{\pm 0.448}$	$33.866_{\pm 0.611}$	$86.210_{\pm 0.129}$	$4.891_{\pm 0.046}$	$31.643_{\pm 0.713}$	$10.234_{\pm 0.019}$	$1.785_{\pm 0.146}$	$8.621_{\pm 0.005}$
CDTD	$2.140_{\pm 0.001}$	$4.578_{\pm 0.216}$	$2.476_{\pm 0.062}$	$4.451_{\pm 0.005}$	$4.313_{\pm 0.040}$	$4.215_{\pm 0.008}$	$4.586_{\pm 0.027}$	$4.303_{\pm 0.003}$	$2.030_{\pm 0.002}$	$4.403_{\pm 0.018}$
XGB-DDPM	$0.425_{\pm 0.011}$	$7.176_{\pm 0.238}$	$7.087_{\pm 0.235}$	$9.921_{\pm 0.217}$	$5.752_{\pm 0.024}$	$1.846_{\pm 0.160}$	$7.617_{\pm 0.202}$	$6.215_{\pm 0.103}$	$0.549_{\pm 0.023}$	$1.997_{\pm 0.147}$

Table 23: **Sampling Time (seconds)** for each dataset and generator. Shows average ± standard deviation over 3 training seeds and 10 sampling seeds. Best scores are **bolded** and second best scores are underlined.

	fiat500	default	credit-g	diamonds	give-credit	wine	bank	jm1	diabetes	shipping
ARF	$0.023_{\pm 0.007}$	$0.585_{\pm 0.528}$	$0.065_{\pm 0.036}$	$0.595_{\pm 0.394}$	$0.441_{\pm 0.450}$	$0.070_{\pm 0.042}$	$0.853_{\pm 0.483}$	$0.155_{\pm 0.142}$	$\mathbf{0.022_{\pm 0.005}}$	$0.173_{\pm 0.100}$
CTGAN	$0.027_{\pm 0.023}$	$0.169_{\pm 0.140}$	$0.042_{\pm 0.025}$	$0.173_{\pm 0.162}$	$0.334_{\pm 0.383}$	$0.039_{\pm 0.006}$	$0.204_{\pm 0.152}$	$0.070_{\pm 0.042}$	$0.033_{\pm 0.028}$	$0.053_{\pm 0.028}$
CTGAN (org.)	$0.034_{\pm 0.053}$	$0.339_{\pm 0.723}$	$0.046_{\pm 0.042}$	$0.212_{\pm 0.150}$	$0.208_{\pm 0.086}$	$0.178_{\pm 0.736}$	$0.235_{\pm 0.097}$	$0.074_{\pm 0.005}$	$0.025_{\pm 0.027}$	$0.051_{\pm 0.003}$
TVAE	$0.024_{\pm 0.015}$	$\mathbf{0.132_{\pm 0.043}}$	$\mathbf{0.039_{\pm 0.018}}$	$0.126_{\pm 0.008}$	$\mathbf{0.110_{\pm 0.005}}$	$0.037_{\pm 0.005}$	$0.197_{\pm 0.133}$	$0.080_{\pm 0.063}$	$0.027_{\pm 0.019}$	$0.053_{\pm 0.020}$
TVAE (org.)	$\mathbf{0.022_{\pm 0.018}}$	$0.179_{\pm 0.360}$	$0.044_{\pm 0.041}$	$\mathbf{0.114_{\pm 0.018}}$	$0.116_{\pm 0.007}$	$\mathbf{0.035_{\pm 0.016}}$	$\mathbf{0.141_{\pm 0.007}}$	$\mathbf{0.055_{\pm 0.010}}$	$0.029_{\pm 0.039}$	$\mathbf{0.037_{\pm 0.003}}$
ForestDiffusion	$0.154_{\pm 0.010}$	$2.829_{\pm 0.100}$	$0.481_{\pm 0.048}$	$4.639_{\pm 0.374}$	$1.487_{\pm 0.050}$	$0.387_{\pm 0.036}$	$3.426_{\pm 0.073}$	$0.665_{\pm 0.063}$	$0.118_{\pm 0.007}$	$0.392_{\pm 0.046}$
ForestFlow	$0.167_{\pm 0.010}$	$1.901_{\pm 0.123}$	$0.600_{\pm 0.098}$	$3.317_{\pm 0.232}$	$0.957_{\pm 0.053}$	$0.338_{\pm 0.035}$	$2.890_{\pm 0.161}$	$0.783_{\pm 0.081}$	$0.132_{\pm 0.008}$	$0.411_{\pm 0.046}$
TabDDPM	$1.656_{\pm 0.012}$	$26.083_{\pm 0.068}$	$3.628_{\pm 0.047}$	$46.369_{\pm 0.169}$	$23.810_{\pm 0.481}$	$5.392_{\pm 0.033}$	$56.174_{\pm 0.344}$	$4.355_{\pm 0.044}$	$0.629_{\pm 0.008}$	$11.360_{\pm 0.078}$
TabSyn	$0.075_{\pm 0.005}$	$0.882_{\pm 0.011}$	$0.075_{\pm 0.007}$	$1.459_{\pm 0.010}$	$1.696_{\pm 0.006}$	$0.183_{\pm 0.007}$	$1.250_{\pm 0.008}$	$0.331_{\pm 0.004}$	$0.370_{\pm 1.645}$	$0.326_{\pm 0.033}$
TabSyn (org.)	$0.109_{\pm 0.004}$	$1.301_{\pm 0.009}$	$0.100_{\pm 0.045}$	$2.175_{\pm 0.009}$	$2.515_{\pm 0.017}$	$0.308_{\pm 0.003}$	$1.852_{\pm 0.008}$	$0.526_{\pm 0.183}$	$0.068_{\pm 0.003}$	$0.478_{\pm 0.014}$
TabDiff	$0.221_{\pm 0.007}$	$1.731_{\pm 0.010}$	$0.525_{\pm 0.010}$	$2.904_{\pm 0.010}$	$2.781_{\pm 0.006}$	$0.437_{\pm 0.004}$	$3.911_{\pm 1.032}$	$0.678_{\pm 0.007}$	$0.226_{\pm 0.006}$	$0.852_{\pm 0.004}$
TabDiff (org.)	$0.566_{\pm 0.008}$	$6.017_{\pm 0.013}$	$1.289_{\pm 0.100}$	$7.218_{\pm 0.094}$	$7.976_{\pm 0.024}$	$1.382_{\pm 0.005}$	$8.163_{\pm 0.118}$	$2.876_{\pm 0.013}$	$0.591_{\pm 0.014}$	$2.110_{\pm 0.022}$
CDTD	$0.170_{\pm 0.005}$	$2.158_{\pm 0.011}$	$0.445_{\pm 0.021}$	$3.648_{\pm 0.016}$	$3.983_{\pm 0.130}$	$0.478_{\pm 0.008}$	$3.889_{\pm 0.012}$	$0.807_{\pm 0.004}$	$0.179_{\pm 0.002}$	$1.007_{\pm 0.004}$
XGB-DDPM	$1.432_{\pm 0.088}$	$9.975_{\pm 0.375}$	$6.029_{\pm 0.177}$	$38.200_{\pm 0.436}$	$3.742_{\pm 0.113}$	$1.951_{\pm 0.149}$	$36.483_{\pm 1.352}$	$4.167_{\pm 0.149}$	$1.507_{\pm 0.017}$	$7.707_{\pm 0.381}$