Under the Hood of Tabular Data Generation Models: Benchmarks with Extensive Tuning

G. Charbel N. Kindji, Lina Maria Rojas-Barahona, Elisa Fromont, Tanguy Urvoy

Under the Hood of Tabular Data Generation Models: Benchmarks with Extensive Tuning

G. Charbel N. Kindji[a,b,*], Lina M. Rojas-Barahona[b], Elisa Fromont[a], Tanguy Urvoy[b],

[a] *Univ Rennes, IUF, Inria, CNRS, IRISA, Rennes, 35000, France*
[b] *Orange Labs, Lannion, 22300, France*

Abstract

The ability to train generative models that produce realistic, safe and useful tabular data is essential for data privacy, imputation, oversampling, explainability or simulation. However, generating tabular data is not straightforward due to its heterogeneity, non-smooth distributions, complex dependencies and imbalanced categorical features. Although diverse methods have been proposed in the literature, there is a need for a unified evaluation, under the same conditions, on a variety of datasets. This study addresses this need by fully considering the optimization of: hyperparameters, feature encodings, and architectures. We investigate the impact of dataset-specific tuning on five recent model families for tabular data generation through an extensive benchmark on 16 datasets. These datasets vary in terms of size (an average of 80,000 rows), data types, and domains. We also propose a reduced search space for each model that allows for quick optimization, achieving nearly equivalent performance at a significantly lower cost. Our benchmark demonstrates that, for most models, large-scale dataset-specific tuning substantially improves performance compared to the original configurations. Furthermore, we confirm that diffusion-based models generally outperform other models on tabular data. However, this advantage is not significant when the entire tuning and training process is restricted to the same GPU budget.

Keywords: Tabular data generation, Generative Models, Evaluation Metrics, Deep Learning, Hyperparameter Tuning, Neural Architecture Search

*Corresponding author. Email: charbel.kindji.orange@gmail.com

1. Introduction

The demand for generative models is rapidly increasing. Particularly, generating realistic, safe, and useful tabular data is crucial for industries, wherein this type of data is most common. Among the direct applications of tabular data generation we can cite data privacy, imputation, oversampling, explainability or simulation [1, 2]. Because of their ability to learn valuable data representations, another significant application of these models is pre-training and fine-tuning for various downstream tasks [3, 4, 5]. However, generating high-quality tabular data presents several technical challenges that are not encountered with text or images [6]. First, tabular columns are encoded through heterogeneous data types with distributions that are often non-smooth with mixed (continuous/discrete) behaviours and various modalities. There can also be complex dependencies between columns, and the categorical features are often highly imbalanced. Finally, the wide variety of problems represented in tabular form makes it challenging to establish a universal data encoding and architecture suitable for pre-training across all scenarios.

To handle these challenges, many models have been proposed in the literature, which are often evaluated on different datasets with inconsistent metrics, tuning and training budgets. Indeed, the performance of the allegedly best tabular generation models seem very unstable from one dataset to another. Moreover, these models seem quite sensitive to the feature-encodings and hyperparameter-choices made by the authors.

In this work we propose a unified evaluation benchmark for tabular data generation. Our main contributions are :

- We present the state-of-the-art and propose the following typology of the existing models : *non-iterative* and iterative neural, which groups *auto-regressive* and *diffusion* models; as well as *non-neural* models.
- We study the impact of dataset-specific feature encoding, hyperparameter and architecture tuning on tabular data generation models. For each model we answer the following questions: (i) is it worth optimizing the hyperparameters/preprocessing specifically for each dataset? (ii) can we propose a reduced search space that fits well for all datasets? (iii) is there a clear trade-off between training/sampling costs, and synthetic data quality?

- We benchmark 5 models that are representative of the recent literature on 16 datasets with a strict 3-fold cross-validation procedure. The datasets were chosen based on their size, purpose and diversity.

- We consider in our benchmark both extensive and limited-budget optimizations for each fold and dataset through hundreds of trials.

We consider multiple metrics for evaluating the models in terms of realism, utility and anonymity as well as costs and carbon footprint. Finally, we analyze the results of these benchmarks and derive some interesting insights about the models. While diffusion-based models like TabSyn and TabDDPM [7] generally outperform other models when left unconstrained, they do not significantly surpass their simpler counterparts when tuning and training budgets are limited. This is because models that do not rely on Transformers have a smaller memory footprint, allowing for more thorough optimization within the same GPU budget.

The paper is structured as follows. First we present the related work in Section 2 and the models' typology in Section 3. The benchmark's challengers, metrics, datasets and other experimental settings are presented in section 4. The extensive and limited-budget benchmarks are detailed in Sections 5 and 6 respectively. Finally the conclusions is presented in Section 7 and the future work in Section 8.

2. Related Work

It is worth noting that this paper is not meant to be a survey. Rather, it benchmarks selected models for tabular data generation through *meticulous fine-grained tuning*. Furthermore, previous surveys on tabular data generation are either focused on privacy [8, 9, 10, 11, 12, 13] or do not cover recent diffusion models [2, 14]. Most survey only report the author's evalautions with default hyperparameters. A few benchmarks like [15, 12] perform hyperparameters search, but with a simple train/validation/test split, a reduced search space, and a small number of trials (usually from 20 to 50). We instead provide large-scale a cross-validated extensive benchmark with up to 300 trial per fold (see Section 4.3). We select representative models of the literature and we evaluate them on distinct datasets in cross validation, in which for each fold and dataset, we optimized the model's hyperparameters,

feature encoding, and architecture through hundreds of trials. Unlike previous work, our benchmarks compare not only the realism, the utility and the anonymity, but also the costs and the carbon footprint.

3. Typology of Models for Tabular Data Generation

Tabular data generation is a booming research field which gives birth every month to a host of new data synthesis algorithms. In this section we survey the existing families of tabular data generation methods with a specific focus on the models that we selected for our study. This includes models already covered in [2] and [14] as well as more recent diffusion-based models and LLM-based ones [16]. We chose models known for their strong performance, widespread usage, and availability of code that can be easily adapted for both architecture and hyperparameter tuning.

The following neural and non-neural approaches have been proposed for tabular data generation. Among the neural approaches, we make a distinction between the "*push-forward*" models which directly map noise into data, and the iterative models which require a decoding phase.

3.1. Non-iterative Neural Models

The most popular non-iterative or "*push-forward*" neural networks for tabular data generation are Variational Auto-Encoders (VAE) [17] and Generative Adversarial Networks (GAN) [18]. A few papers also consider self-normalizing flows [19]. One of the simplest VAE architecture for tabular data is TVAE [20]. In its original implementation [21], it consists of a one-hot encoding for categorical variables coupled with a Gaussian Mixture Model normalization scheme (GMM) for continuous features. The encoder/decoder architecture is a simple stack of linear layers. Several variants have been proposed to improve from this baseline. In [22] the GMM normalization is replaced by a two-step training that first fits the marginals then fits the inter-dependencies. In [23, 24], several normalization schemes were tested as a replacement for the GMM normalization. Other variants of VAE use differentiable oblivious trees to ensure privacy [25]. In [26] the VAE is coupled with a Graph Neural Network (GNN).

The most popular method to generate tabular data is certainly adversarial training [27, 28, 29, 30, 31, 20, 32, 23, 33, 14, 34]. It would not be exaggerated to affirm that every exotic variant of GANs has been tested on tabular data generation, but the most successful architectures seem to be

the ones based on Wasserstein GANs [35] such as CTGAN [20]. CTGAN is the base architecture that we selected for our benchmark. As mentioned in [23, 24], the feature encoding scheme is critical, especially for numerical features. For this reason we customized the CTGAN code as we did for TVAE, to allow for a choice of architecture and feature encoders.

3.2. Iterative Neural Models

It has been shown that iterative generative models (either auto-regressive or by diffusion) almost systematically outperform push forward models when it comes to raw text, sound, or image generation [36]. We confirm here that tabular data do not escape this rule. However, this performance comes with a cost: the decoding phase is often slow and highly energy consuming.

3.2.1. Auto-regressive Language Models

After the recent breakthrough of large language models (LLM) [37], the usage of token-based language models to generate tabular data seems inevitable. Their main advantage against prior *ad-hoc* models is that they come without specific feature encodings for numerical or categorical columns: these values are directly fed to the model as raw sequences of tokens. Even if there is no clear agreement yet on how tabular instances should be serialized for LLMs, this general encoding ability opens the possibility for a pretraining/finetuning paradigm on heterogeneous tabular datasets [3, 5].

In a recent preprint survey [16], the authors tried to map the exuberant flow of preprint papers on tabular data and LLMs. Several of these preprints present only prompt engineering tricks that generate small tables, nonetheless some of the proposed models seem promising at a larger scale. To name a few of them, GReaT [38] (for "Generation of Realistic Tabular data") proposes to fine-tunes GPT-2 [39] for tabular data generation. REaLTabFormer [40] extends GReaT to multiple tables with shared indexes, and TAPTAP [5] experiments a pretraining of GReaT on 450 open tabular datasets.

We made a few experiments with GReaT and its variants and confirmed the remarks of [7, 12], which state that they struggle to capture the joint probability distribution on datasets where the categorical values names do not carry semantic information. Given the prohibitive computational cost of LLMs, the large number of hyperparameters and serialization schemes to consider, as well as the problematic fact that some datasets are already covered (i.e. used in the training data) by the foundation models training sets, we decided to postpone their evaluation for a future work.

3.2.2. Diffusion models

Another recent impressing breakthrough in generative modeling, especially in image generation, was the apparition of diffusion models [41, 42, 43, 44, 45]. These models learn to iteratively transform random Gaussian noise to a sample from an unknown data distribution. This transformation is defined through a forward and a backward diffusion processes. The first one iteratively adds Gaussian noise to samples, while the second one iteratively "denoises" samples from the Gaussian distribution to obtain samples from the data distribution. The transposition of these models to tabular data gave rise to powerful synthesizers: TableDiffusion [46], Stasy [47], CoDi [48], and TabDDPM [15]. In TabSyn [7], a more recent proposal, the authors transposed the idea of [49, 50] to make use of a transformer-based VAE in order to embed the diffusion in a latent space. We selected both TabDDPM and TabSyn for our benchmark.

3.3. Non-neural Models

The most common statistical approaches are based on *copulas* and *Probabilistic Graphical Models* (PGMs). Copulas [51] are functions that join or "couple" multivariate distribution functions to their one-dimensional marginals. They have been widely adopted for tabular data generation because they allow modeling the marginals and the feature inter-dependencies separately [21, 52, 53, 54, 55]. Nevertheless, parametric copulas have been shown to perform poorly on high-dimension data synthesis problems [20, 24].

On the other hand PGMs can model variable dependencies in high-dimension spaces [56, 57, 14, 58, 59, 60]. This approach has been shown to be quite efficient in the data privacy community [61, 10, 11, 12]. However, it often requires a prior knowledge on the dependency graph because graph inference from data is inefficient in high dimension, especially if the sample size is small [62].

The fact that ensembles of trees remain state of the art for predictive tasks on tabular data [6] motivated some interesting attempts to mimic the neural generative approaches with decision trees. It gave rise to adversarial forests [34], Forest-Flow, and Forest-VP (a variance-preserving diffusion algorithm) [63]. We tested Forest-VP but we encountered a severe scalability issue: contrary to neural diffusion models, introduced in Section 3.2.2, where the noise level is combined with the input as an auxiliary variable, the Forest-VP algorithm trains a different ensemble of trees for each level of noise.

Another simple, but quite efficient way to generate new tabular data is to interpolate between existing instances. This geometric "nearest neighbors" approach called *Synthetic Minority Over-sampling Technique* (SMOTE) [64, 65, 66] is frequently used to resample instances before training predictive models on unbalanced datasets. It proceeds by picking a random instance with a fixed target value and finding its k nearest neighbors. New data points are then generated by interpolation between these neighbors. Although very simple, this model is a solid baseline for tabular data generation as shown in [15].

4. Benchmarks Settings

We first present the selected challengers and the various evaluation metrics. Then, we present the optimization framework that we developed and we discuss the choices made to wrap the different challengers into this framework. Finally, We present the datasets.

4.1. Selected Challengers

We selected five models for our benchmarks, two iterative neural: TVAE, CTGAN; two diffusion models: TabDDPM, and TabSyn; and one non-neural: SMOTE (with a variant unconditional SMOTE, namely ucSMOTE). We followed [24] and used a customized version of TVAE which allows for an optimized choice of the architecture and of the feature encoder.

For each of these algorithms we had to carefully examinate the code to optimize large scale hyperparameters, features encoding and architecture. We also reported the trivial baseline which consists in resampling directly the train set, namely "*Train Copy*".

4.2. Evaluation Metrics

Our purpose is to assess the quality of tabular data generation though multiple facets. These facets can be summarized with four questions: (i) is synthetic data realistic? does it respect the original distribution's traits? (ii) is it useful? *i.e.* can it be used to train machine learning models? (iii) do synthesis preserve training data anonymity? does it overfit? (iv) what are the model's costs and CO_2 impacts? Each of these questions is related to specific metrics.

The first and most important question is the realism of the generated data. We rely on *Classifier Two Sample Test*[1] (C2ST) [67] as a primary metric to address it. This metric is also the one that we used for hyperparameter optimization. It evaluates the performance of a classifier at discerning real data from synthetic data[2]. To compute C2ST we use the same protocol as in [24] where the computed value is the mean ROC-AUC of XGBoost [68] over three folds. A C2ST around 1/2 means that XGBoost is unable to discern the test set from the generated set. A high C2ST means on the contrary that XGBoost is able to detect easily the fake data. The C2ST calculation procedure is summarized in Figure 1.

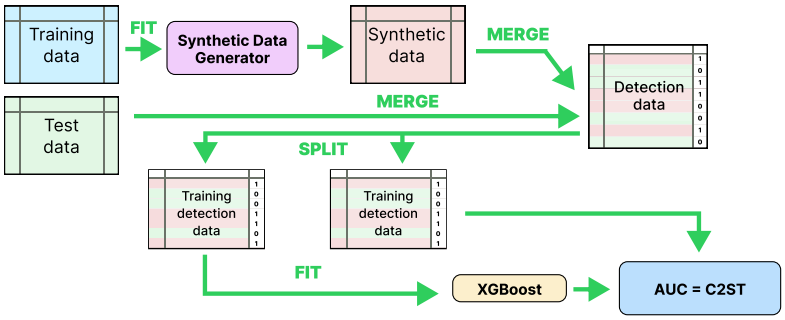


Figure 1: C2ST Metric calculation.

We also consider two other statistical metrics for data realism: *column-wise similarity* and *pair-wise correlation*. To compute these metrics we used the SDMetrics library [21]. The column-wise similarity measures how accurately the synthetic data captures the shape of each column distribution individually. It is reported as "Shape" in the result tables. Pair-wise correlation on the other hand captures how each column varies with each other. Pair-wise correlation is reported as "Pair" in the result tables. A naive generator that would assume independence of the columns might have a high "Shape" score but it would have a low "Pair" score.

The second important question is the utility of generated data. It is commonly measured by ML-Efficacy which evaluates the performance of a predictive model trained on synthetic data. To compute ML-Efficacy we use

[1] Also mentioned as *Detection test* in SDMetrics `https://docs.sdv.dev/sdmetrics`

[2] This is similar to what is done in GANs but with a fresh dataset. An overfitted GAN can indeed have a poor C2ST on test set.

the same protocol as in [15] where CatBoost [69] is used as a predictor. We report the F1 score for classification tasks and the normalized R2 score for regression tasks. The procedure is summarized in Figure 2. It is important to evaluate the degradation of these scores against the ones obtained when training directly on real data (via *Train Copy*): a large degradation means a low utility.

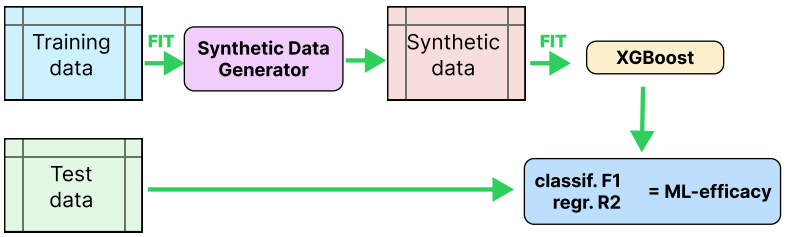


Figure 2: ML-Efficacy Metric calculation.

The third question is anonymity. We want to make sure that our generative model will not leak sensitive information by recopying or over-fitting the training instances. To do so we measure respectively the minimum distances of each generated instance to the train and test sets. The *Distance to Closest Record* Rate (DCR-Rate) counts the proportion of generated instances that are closer to train set than test set [70]. The procedure is summarized in Figure 3.

A synthetic dataset is considered safe if it has a DCR-Rate that is close to 1/2. On the other hand, a plain copy of the train set, as *Train Copy* baseline does, would have a DCR-Rate of 1. This metric does not guarantee against all privacy breaches, but it provides a reasonable safeguard and ranking criterion for the models. It is also informative about overfitting, as an overfitted model would have samples that are systematically closer to the train set than a model with more generalization capabilities.

The fourth question is the models' costs and their carbon impacts. It requires an accurate estimate of three cost values: *time*, *energy consumption*, and CO_2 *impact*. Ideally, these values should be estimated for the three phases of (i) training (gradient descent) (ii) sampling, and (iii) hyperparameters search. For each dataset, each fold and each optimized model architecture, we ran the training and sampling phases on the exact same hardware and software

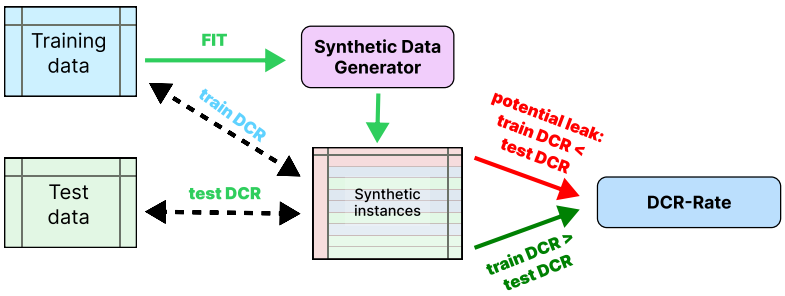


Figure 3: DCR-Rate Metric calculation.

architecture, a single Tesla V100 32 GB, and we measured accurately the cost values with the *CodeCarbon* library[3]. However, due to the massive nature of the experiments, we could not perform the whole hyperparameter search and training phases on such a uniform hardware and software architecture. We hence estimated the global search costs from the tuning logs by rescaling the training cost measures according to the effective number of steps performed and the number of GPU used (c.f. equation (1)).

$$\text{total-gpu-cost} \simeq \sum_{t \in \text{trials}} \frac{\text{init-cost}_t + \text{avg-cost-per-step} \times \text{num-steps}_t}{\text{trials-per-gpu}} \qquad (1)$$

This is slightly overestimated since: (i) *CodeCarbon* assumes a 100% GPUs load while ours was roughly around 95%; and (ii) we measured the step costs on already optimized models which are usually slower because they often count more layers and parameters. Note that the number of trials that can be parallelized on a single GPU depends on the memory footprint of the model[4].

4.3. Large Scale Optimization Framework and Implementation Details

Providing a fair and reliable comparison of the different tabular generative models is a tough technical challenge. For this reason most existing benchmarks like [38, 7] only report the performance of models with their default hyperparameters. A few benchmarks like [15, 12] perform hyperpa-

[3]https://codecarbon.io/

[4]For instance we could safely run 10 TVAE trials on the same V100 GPU while only 4 TabSyn's trials could fit because of the VAE transformer's footprint.

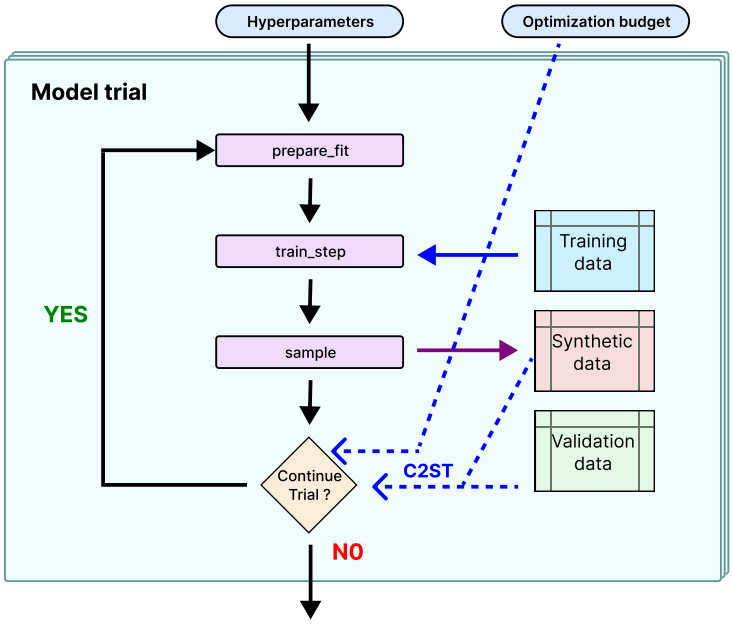


Figure 4: Hyperparameters trial optimization loop.

rameters search for all models, but with a simple train/validation/test split, a reduced search space, and a small number of trials (usually from 20 to 50).

We wanted our experiment to be more extensive and more robust, so we decided to deploy it at a large scale on a super-computer[5] equipped with several nodes with 4 GPUs V100 32GB each[6].

For this purpose, we used the *Ray tune* distributed library [71] coupled with *Hyperopt* [72] which is based on Tree Parzen Estimators (TPE) [73] and Asynchronous Successive Halving Algorithm (ASHA) as a scheduler [74] to optimize hyperparameters and architecture efficiently. Depending on the model's memory footprints, each GPU could host from four to twelve concurrent tuning trials. For each dataset and each model we performed a strict 3-fold cross validation, which means that for each tuple (dataset, fold, model) we performed an extensive hyperparameters search with 300 trials (except for TabSyn where we reduced this number to 100 for technical reasons explained

[5]Jean-Zay super-computer `www.idris.fr`

[6]Complementary experiments were also performed on a 2×RTX4090 workstation.

in Section 4.3.2). We hence obtained a different optimized architecture for each (dataset, fold, model) tuple.

In [15] the parameters were optimized for ML-efficacy, in [12] the parameters were optimized for an equal combination of realism, utility and privacy. We chose to optimize for realism as in [24] through XGBoost-based C2ST metric (c.f. Section 4.2). To obtain reliable evaluations with variance estimates, for each dataset we evaluated the models by averaging all the metrics on the three folds test sets with five synthetic samples for each fold.

As described in Figure 4, in order to work with our framework each algorithm has to be wrapped into a generic *Synthesizer* class that provides three methods: *prepare_fit* which prepares the dataset and the model according to the hyperparameters, *train_step* which performs a training step roughly equivalent to one or a few epochs, and *sample* which generates synthetic data. After each train step, the model trial was evaluated and it was canceled out by early stopping or by the Ray-Tune scheduler if it performed too poorly or if the time budget was depleted.

4.3.1. The importance of feature encoding

Table 1 presents the encoding schemes that we used for the different benchmark challengers. As pointed out in [6, 23, 24], categorical variables are not the main weakness of neural networks but numerical feature encoding is critical. Most recent neural models use a Quantile-based numerical feature encoding and seem to work well with it. However, the original versions of TVAE and CTGAN rely on a specific Cluster-Based normalization [20]. We hence explored several encoding policies for these two models through hyperparameters optimization (see Tables A.6 and A.7 in the appendix section).

We kept the native Cluster-Based[7] encoder as described in [20], along with the ones proposed in [24], namely: prototype encoding (PTP) [24], piece-wise linear encoder (PLE) [75], continuously distributed residuals (CDF) [76, 24], hybrid (PLE_CDF) [24]. We also added some standard *scikit-learn* transformers: MinMaxScaler and QuantileTransformer.

Contrary to QuantileTransformer which maps values deterministically, the CDF encoding uses randomization to produce continuously distributed residuals even when the original distribution is discrete or partially discrete [76]. The PLE encoder [75] performs a feature binning and normalizes each

[7]https://docs.sdv.dev/rdt/transformers-glossary/numerical/clusterbasednormalizer

Model	Num. Encoder	Cat. Encoder	Num. Target
TVAE_base	Cluster-Based [7]	One hot	-
CTGAN_base	Cluster-Based[7]	One hot	-
TVAE	*Optimized*	One hot	-
CTGAN	*Optimized*	One hot	-
TabDDPM	Quantile	One Hot	Standardize
TabSyn	Quantile	Embedding	-
SMOTE	-	One hot	Median cut
ucSMOTE	-	One hot	Dummy

Table 1: Encoding schemes applied to TVAE and CTGAN.

numerical value depending on the bin it belongs to. The PLE_CDF applies a CDF to the output of a PLE encoding. The PTP encoding, inspired by prototypical networks [77], encodes the input as a weighted average of fixed prototypes.

4.3.2. Model-specific implementation details

Wrapping heterogeneous models within a *Synthesizer* class required some implementation choices from our part, and despite all our efforts to keep a fair comparison, these choices had some impact on the compute time and performance of the models.

The first issue is the discrepancy of the training step's costs. The usual training step unit for most models is the *epoch* which corresponds to a single pass on all instances of the training set. However, depending on the hyper-parameters some models like CTGAN perform both one pass through the generator and several passes through the discriminator at each training step. Other models such as TabDDPM are randomized and require several quick passes (almost one for each level of noise) for each instance. We hence had to caliber our wrapper's train_step functions to perform a compute effort that is roughly equivalent to an epoch.

Another issue is the fact that TabSyn [7] combines two models and was not designed for hyperparameters tuning. According to the authors the model does not need hyperparameters tuning[8]. We decided to train a new

[8]Our experiment show however (c.f. Section 5), that even if the non-optimized TabSyn is quite good on most datasets, the hyperparameter tuning clearly improves the quality of

transformer-based VAE for each trial because it hosts most of the parameters, compute-time and hyperparameters of TabSyn. Training a diffusion model on an unstable latent space would not be meaningful. We thus considered three technical options: (i) optimize first the VAE on a proxy metric (for instance the ability of its decoder to generate realistic data from a standard Gaussian), then optimize the denoiser in the latent space; (ii) wrap the VAE training steps into the *train_step* function and retrain a new denoiser from scratch at each step; (iii) wrap the VAE training phase into the *prepare_fit* function and loose the ability to prune its training steps. The first option is the cheapest and it is probably recommended for most practical applications, but it may be sub-optimal due to the proxy metric. The second option is extremely costly. We hence opted for the third option although it had a non-negligible cost. Indeed, we followed the recommendation of [7] to train the VAE through 4000 epochs which turns out to be huge knowing that most other models only utilized 400 epochs in our benchmark.

We also reduced the number of parallel trials per GPU because of the large memory footprint of the VAE's transformers. As a consequence, for TabSyn we had both to reduce the number of trials to 100, and to work on a sample of the largest dataset (*Covertype*) to get the results in a reasonable amount of time. It is worth noting that despite these handicaps, the optimized TabSyn remained better than its non-optimized version. To avoid this experimental bias for the second experiment in Section 6.2, we constrained TabSyn's VAE to use only 10 minutes for training. On *Adult* dataset, for instance, it resulted in roughly 560 epochs.

Finally, the last issue was the different ways the models deal with the target columns. A model conditioned on the target column may improve its ML-efficacy. TabDDPM and SMOTE implementations are natively conditioned on classification targets, while TVAE, CTGAN and TabSyn are not.

We did not modify TabDDPM, but we considered two variants in our experiment for SMOTE: the first, that we call SMOTE, follows the design of [15], it uses the train target distribution for classification datasets and a rough median split for regression targets (see Table 1). The second, that we call ucSMOTE (for unconditional SMOTE), adds a dummy target filled with zeros to the data before calling the SMOTE oversampling library. By doing so, all columns, including the original target, are considered equally.

---

the data it generates.

4.4. Datasets

To evaluate the models, we picked datasets with various characteristics to assess their performances under different scenarios. Datasets were chosen to cover various sizes, dimensions, different types of tasks (regression, binary, and multi-class classification) and various types of features (numerical, categorical, or mixed). We also added *Moons* a well known *scikitlearn* synthetic dataset. The complete list of datasets and their characteristics is presented in Table 2. The *Covertype* dataset size was reduced for TabSyn tuning as follows: 27500 in the training set, 18333 in the validation set, and 9167 in the test set.

Name	Train	Validation	Test	Num	Categ.	Task
Abalone[9]	2088	696	1393	7	2	Binclass
Adult[9]	24420	8141	16281	6	9	Binclass
Bank Marketing[9]	22605	7535	15071	7	10	Regression
Black Friday[9]	83410	27804	55607	6	4	Regression
Bike Sharing[9]	8689	2897	5793	9	4	Regression
Covertype[9]	290505	96836	193671	10	45	Multiclass
Cardio[10]	34999	11667	23334	11	1	Binclass
Churn Modelling[10]	4999	1667	3334	8	4	Binclass
Diamonds[9]	26970	8990	17980	7	3	Regression
HELOC[10]	5229	1743	3487	23	1	Binclass
Higgs[9]	49024	16342	32684	28	1	Binclass
House 16H[9]	11391	3798	7595	17	0	Regression
Insurance[10]	669	223	446	4	3	Regression
King[10]	10806	3602	7205	19	1	Regression
MiniBooNE[9]	65031	21678	43355	50	1	Binclass
Two Moons	19999	6667	13334	2	1	Binclass

Table 2: List of datasets. Direct links to exact versions of datasets used can be found in Appendix B.

5. Extensive Benchmark

As mentioned in Section 4.2, the evaluation and comparison of tabular generative models is based mainly on four criteria: realism, usefulness,

[9] `https://www.openml.org`

[10] `https://www.kaggle.com/datasets`

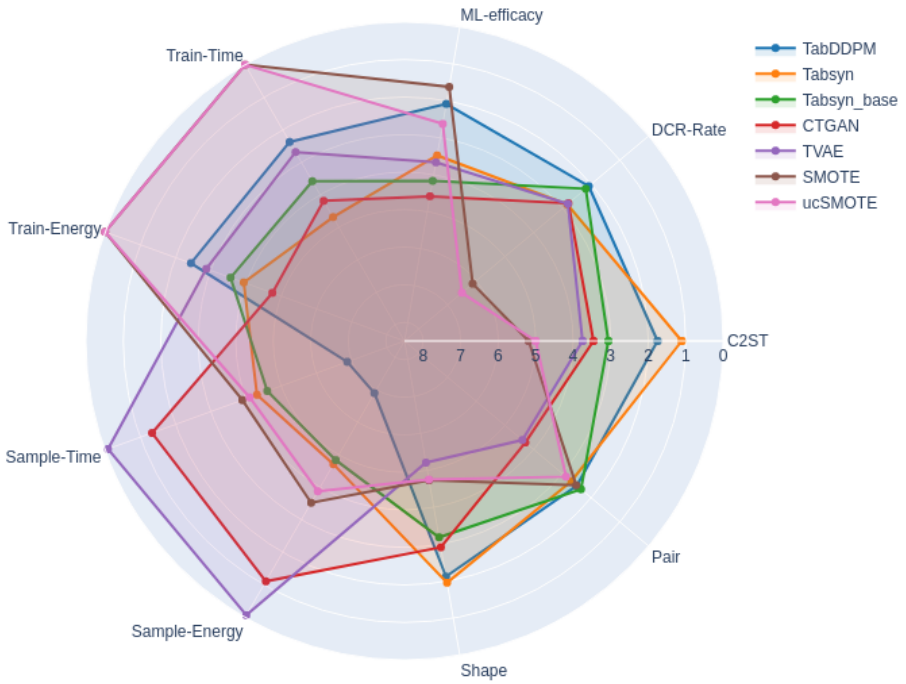


Figure 5: Radar chart of the **extensive experiment** showing optimized model's average ranking on all datasets across various metrics. The training costs of optimized models include both tuning and gradient descent.

anonymity, and cost. After a global multi-criteria overview of the results, we study and compare the model's behaviour according to each criterion individually.

5.1. Multi-criteria Overview

Figure 5 shows the average ranking over all datasets and folds of seven model variants over eight metrics (c.f. Section 4.2). To complete these rankings, Table 3 and Table 4 summarize respectively the quality metric and cost distributions among all datasets and folds. Recall that the best scores for the C2ST are around 0.50 (as it means poor AUC for the classifier at telling synthetic data apart from holdout data).

Overall, no model provides the best performance over all considered criteria. We observe, as expected, a strong correlation between energy and GPU time as well as a strong correlation among "quality metrics" (*i.e.* C2ST, *Shape*, and *Pair*). On the one hand, the diffusion models achieve the best

performance in terms of quality metrics, especially the tuned version of TabSyn. As shown in Table 3, this model has a median C2ST value of 0.64. However, TabSyn is also one of the most expensive models in term of training costs (*Train-Energy* as well as *Train and Sample Times*), as it requires to train both a transformer-based VAE and a denoiser model for each dataset.

On the other hand, the SMOTE baselines obtain the poorest quality and privacy ranking with a high DCR-Rate. The median DCR-Rate score for SMOTE and ucSMOTE is at 0.97 which means that most of the samples from these models are very similar to the training set. However, they achieve strong utility in terms of *ML-Efficacy* with quartiles very close to *Train Copy* and the same median value of 0.73. As expected for neighborhood-based algorithms, their training cost is negligible, but their deployment requires a neighborhood search which can be costly on large datasets. Finally the tuned neural push-forward models (CTGAN and TVAE) achieve mitigated results in term of both quality and utility but their deployment is clearly the cheapest. We also note that all neural models achieve reasonable results in term of privacy preservation. Although TabDDPM is clearly the slowest algorithm for deployment, it is fast at training and it obtains homogeneous results over all other metrics. We note that the tuned version of TabDDPM is performing better than the base version of TabSyn.

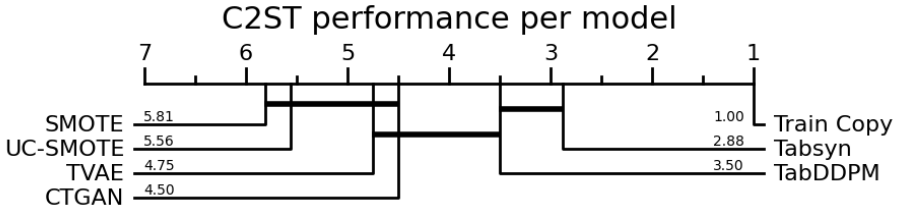
For this extensive experiment we only limited mildly the time budget and the number of epochs. However, some models like TabSyn and CTGAN consumed much more GPU time than others, especially TabDDPM which was very quick at performing an equivalent number of epochs (c.f. Section 4.3.2). It is also important to compare these algorithms with a fair allocation of GPU resource as we do in Section 6.

5.2. Detailed Analysis

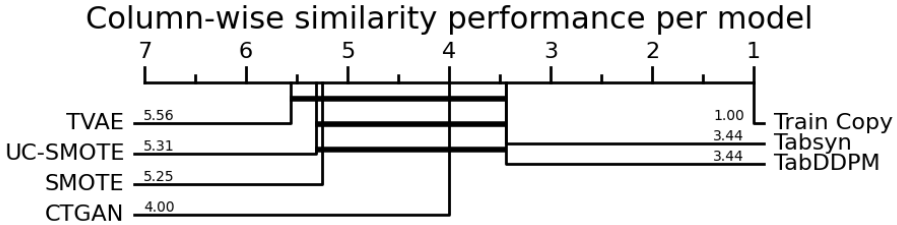
In this section we study and compare the model's behaviour according to each criterion taken individually. For each dataset we computed the performance metrics over 3 folds and 5 synthetic samples per fold to provide a stable central tendency and dispersion estimate. The full dataset-level results are provided in Appendix Table C.12. We summarize these results among all datasets and folds in Table 3 and Table 4. For quality metrics we compare the models through critical difference diagrams.

Model	Percentiles	C2ST ↓	DCR-R ↓	ML-EF ↑	Shape ↑	Pair ↑
Train Copy	75%	0.50	1.00	0.90	0.99	0.99
	50%	0.50	1.00	0.73	0.99	0.98
	25%	0.50	1.00	0.63	0.99	0.91
TVAE	75%	0.88	0.65	0.79	0.96	0.95
	50%	0.81	0.63	0.71	0.95	0.92
	25%	0.74	0.61	0.50	0.94	0.85
TVAE_base	75%	1.00	0.62	0.70	0.92	0.90
	50%	0.98	0.61	0.65	0.91	0.80
	25%	0.96	0.60	0.44	0.87	0.74
CTGAN	75%	0.90	0.64	0.71	0.98	0.96
	50%	0.83	0.63	0.69	0.97	0.91
	25%	0.71	0.61	0.47	0.96	0.85
CTGAN_base	75%	1.00	0.62	0.63	0.92	0.93
	50%	0.99	0.60	0.47	0.88	0.83
	25%	0.93	0.60	0.30	0.87	0.76
TabDDPM	75%	0.78	0.64	0.80	0.98	0.98
	50%	0.67	0.62	0.69	0.98	0.93
	25%	0.63	0.61	0.58	0.95	0.84
TabDDPM_base	75%	0.86	0.64	0.75	0.99	0.96
	50%	0.77	0.62	0.68	0.98	0.92
	25%	0.65	0.61	0.44	0.94	0.74
TabSyn	75%	0.80	0.63	0.76	0.99	0.97
	50%	0.64	0.62	0.68	0.97	0.93
	25%	0.59	0.61	0.46	0.96	0.74
TabSyn_base	75%	0.86	0.64	0.75	0.98	0.97
	50%	0.71	0.62	0.57	0.97	0.95
	25%	0.63	0.61	0.29	0.95	0.88
SMOTE	75%	0.97	0.98	0.86	0.97	0.99
	50%	0.90	0.97	0.73	0.95	0.95
	25%	0.80	0.86	0.60	0.93	0.85
UC-SMOTE	75%	0.95	0.98	0.87	0.97	0.98
	50%	0.88	0.97	0.73	0.95	0.95
	25%	0.81	0.91	0.59	0.93	0.87

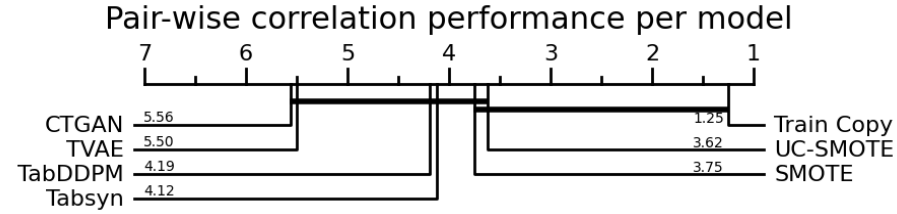
Table 3: Summary results from the ***extensive*** benchmark and the ***base*** models (i.e. with default hyperparameters), DCR-R stands for DCR-rate and ML-EF for ML-Efficacy.



(a) C2ST



(b) Column-wise shape similarity



(c) Pair-wise correlation

Figure 6: Models ranking with critical difference diagrams for C2ST, pair-wise correlations, and column-wise metrics over all datasets.

5.2.1. Are synthetic data realistic ?

We show in Figure 6 the *critical difference diagrams* [78] of all tuned models respectively for C2ST, pair-wise correlation and column-wise similarity. These diagrams were obtained by aggregating the ranks of the seven models over all datasets and folds. A thick horizontal line groups the set of models for which the pairwise "*no significant difference*" test hypothesis could not be rejected.

If we except the trivial *Train Copy* policy which is by construction the most realistic generator, we note that TabSyn is significantly better in terms

of C2ST than all other models except TabDDPM. On the other side, the SMOTE baselines are significantly worse than TabDDPM and TabSyn. The absolute C2ST values in Table 3 corroborate these ranking results with three *sets* of models: (i) diffusion-based models TabSyn and TabDDPM with a C2ST lower than 0.80 on most datasets and median around 0.65; (ii) push-forward models CTGAN and TVAE with a median C2ST around 0.82; (iii) SMOTE algorithms with a C2ST higher than 0.80 on most datasets.

The rankings obtained according to the *column-wise similarity* are broadly the same as the one obtained for C2ST. In addition to the rank shown in Figure 6(b), we can see that all the models obtain good scores (usually around 0.96) as compared to the *Train Copy* baseline (0.99). This result suggests that all the models succeed at capturing univariate distributions. For *pair-wise correlation*, we note surprisingly high values for SMOTE and ucSMOTE baselines while the neural network models obtain broadly the same ranking as for C2ST but the gaps between models are less marked.

A side takeaway from this result is that XGBoost-based C2ST provides a stronger discriminative power than *column-wise similarity* and *pair-wise correlation* metrics.

5.2.2. Can the synthetic data be used to train a machine learning model ?

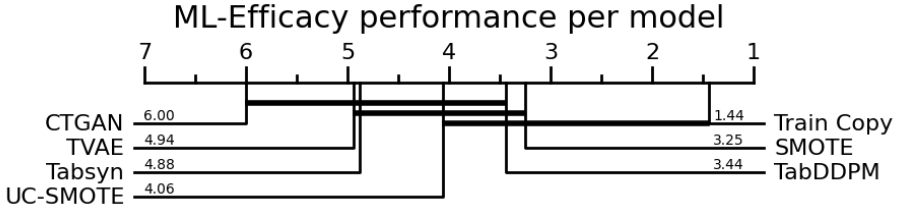


Figure 7: Models ranking with critical difference diagram for Catboost ML-Efficacy over all datasets.

According to the *machine learning efficacy* metric (ML-Efficacy), the most useful generators are the ones that are conditioned on their targets (namely SMOTE and TabDDPM) with median values respectively at 0.73 and 0.69 (against 0.73 for *Train Copy*).

TabDDPM outperforms both base and tuned versions of TabSyn for this metric. It is also safer than SMOTE and it performs its training iterations faster than the other evaluated models. If ML-Efficacy is of importance, it

is advisable to use this model. As expected, all models are far from the performance obtained on real data (*Train Copy*).

5.2.3. Does synthetic data preserve anonymity?

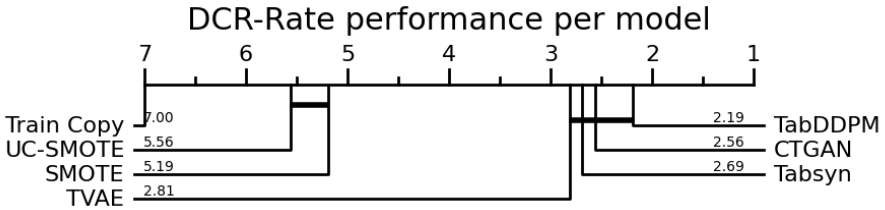


Figure 8: Models ranking with a critical difference diagram for the DCR-Rate metric over all datasets.

A data synthesizer that would only copy its training set would be of little value. If it generates new instances that are too close to its training set, it would obtain a good C2ST score, but it would be prone to over-fitting and it would leak private information from the training set. We assess the ability of a model to generate new data through the DCR-Rate metric (c.f. Section 4.2).

On Figure 8 we observe two significantly distinct groups of models. On the left-hand side a "leaky" group that contains both SMOTE and ucSMOTE, and on the right-hand side, a "safe" group that contains all neural algorithms. The poor performance of SMOTE is mainly due to the way it generates new data points by interpolating between existing ones. Therefore, these models cannot be considered safe concerning data protection. By taking a look at Table 3, the DCR-Rate of the two SMOTE variants is almost always above 0.86. On the other hand, if we exclude the tiny "Insurance" dataset where CTGAN and TVAE overfitted, the DCR-Rate values of the "safe" group are quite uniform around 0.62 and almost always below 0.65: these models can be considered safe.

5.2.4. What are the models' costs?

Model training, optimization, and data sampling have a cost and an environmental impact that varies greatly from one model to another. We hence measured and estimated time, energy consumption, and CO_2 impact for each model during three phases: (i) training (measured), (ii) hyperparameters search (estimated), and (iii) sampling (measured). These results

Model	Percentiles	Duration (HH:MM) ↓	Emission (Kg) ↓	Energy (kWh) ↓
TVAE	75%	01:00	1.05	15.57
	50%	00:24	0.42	6.22
	25%	00:12	0.21	3.08
CTGAN	75%	02:20	2.45	36.36
	50%	01:37	1.67	24.84
	25%	01:06	1.16	17.14
TabDDPM	75%	00:26	0.36	5.30
	50%	00:18	0.29	4.37
	25%	00:12	0.19	2.82
TabSyn	75%	03:25	1.92	22.65
	50%	01:59	1.03	14.20
	25%	01:29	0.72	9.09
SMOTE	75%	00:02	0.00	0.01
	50%	00:00	0.00	0.00
	25%	00:00	0.00	0.00
UC-SMOTE	75%	00:03	0.00	0.01
	50%	00:00	0.00	0.00
	25%	00:00	0.00	0.00

Table 4: Summary of the *costs* of the ***extensive*** benchmark. *Costs* dispersion of the 25th, 50th, and 75th percentiles are provided over all datasets and folds.

are reported fully in Appendix D. We summarize the tuning and training process cost distributions over all datasets and fold in Table 4. As mentioned in Section 4.2, these values are estimated from the tuning logs by taking into account the effective number of training steps, the number of trials, and the average GPU resource usage per training step as reported in Table D.13.

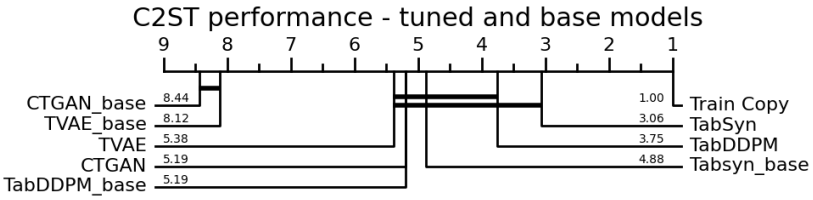
With a median tuning time around 18 minutes as shown in Table 4, TabDDPM is the fastest neural model. As a result, it also consumes less energy and it has less emissions at the training stage. As shown in Figure 5, considering its other performance metrics, it is a suitable choice to achieve good results at a relatively low training cost.

As expected, Figure 5 shows that the push models CTGAN and TVAE are the fastest at sampling stage. We notice that although CTGAN achieves slightly better quality results than TVAE, it is also one of the costliest models at the training stage as shown in Table 4. In order to achieve reasonable performance while reducing training costs, TVAE is hence an option to consider prior to CTGAN.

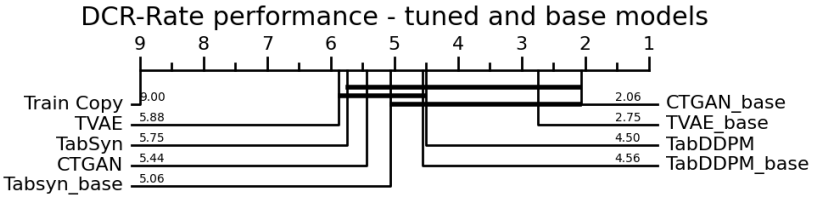
The training and tuning of TabSyn is one of the most demanding (with a median time of of 2 hours for tuning). In the end, however, it delivers the best performance in terms of quality metrics. This model also has the advantage of providing a set of default hyperparameters that can have a *reasonable* performance although, if we follow the author's recommendation of 4000 VAE epochs, its training remains costly by comparison to other models. Indeed, Figure 5 show that, even if we consider the whole tuning+training pipeline TabDDPM and TVAE remain cheaper to train than TabSyn_base.

In terms of sampling cost, TabDDPM is the worst-performing model. It takes longer than the other models (c.f. Figure 5) and hence consumes more energy with more CO_2 emissions at this step. TabSyn reduces the number of denoising steps by using VAE embedding and linear noises to reduce its sampling time [7]. It hence achieves better performance at inference than TabDDPM.

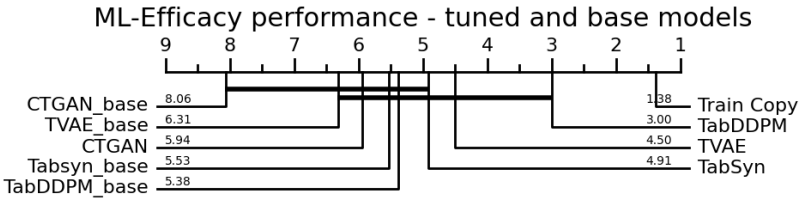
The two baselines SMOTE and ucSMOTE being based on neighbourhood interpolation their training and tuning cost is negligible. However, as shown in Appendix Table C.12, their sampling process requiring a nearest neighbor search, it is slower on large datasets than push-forward models like TVAE or CTGAN.



(a) C2ST



(b) DCR-Rate



(c) ML-Efficacy

Figure 9: Models' ranking with critical difference diagrams on C2ST, DCR-Rate, and ML-Efficacy metrics over all datasets: base models (i.e. with default hyperparameters) versus extensively tuned models.

5.3. Is it worth optimizing the hyperparameters for all models ?

As mentioned in the previous section, even with the help of sophisticated search algorithms, hyperparameter tuning is costly and the performance-versus-tuning-budget curve is following a *diminishing returns law*. We were hence interested in comparing heavily tuned models against non-tuned models.

To assess this, we trained the neural models using the hyperparameters provided by the authors in the original papers. These results are detailed in Appendix E.

In Figure 9(a) we can see a, sometimes huge, C2ST performance improvement of all models when tuned. This improvement is statistically significant between optimized TVAE and CTGAN and their base versions. Looking at

the absolute C2ST values in Table 3, it confirms that these models should not be used with their default hyperparameters.

For TabDDPM and TabSyn, we also notice a 10 points improvement on the median C2ST but this gap is not large enough to provide statistical guarantees.

Although it was not the main target for tuning, we also observe an ML-Efficacy gain for all models. This gain is marked at the 25^{th} percentile (*i.e.* for the hardest datasets).

We can therefore conclude that for all neural tabular generative models that we considered (including TabSyn), it is worth optimizing the hyperparameters specifically for each dataset if we want to improve performance. But the trade-off between the optimization cost and the performance gain is highly correlated to the size and design of the hyperparameter's space (c.f. Appendix A). In the next section we propose a reduced search space and study the impact of a "light" hyperparameters optimization with a limited budget.

6. Limited-Budget Benchmark

As underlined in Section 5.3, optimizing the hyperparameters for each dataset can significantly improve the quality of the generators. But a large-scale optimization like the one we performed is technically difficult, costly, and it has a non-negligible carbon impact (c.f. Table 4). Researchers and practitioners may hence be interested in reducing this cost without deteriorating too much the model's performance.

In this section, we leverage the results of our extensive tuning experiment to: (i) suggest reduced search spaces achieving reasonable performance at a much lower cost; (ii) assess and compare the performance of the models when tuned and trained with the same limited budget. By comparing the model's performance after this limited-budget tuning/training with our previous results (respectively with heavy tuning or without tuning at all), we gain new insights into the models.

6.1. Hyperparameters Search Space Reduction

We carried out this experiment for the most hyperparameter-heavy models, namely: TVAE, CTGAN, TabDDPM, and TabSyn. To reduce their search space, we independently considered each hyperparameter/architecture configuration variable and kept only the values that were the most frequently

selected during the large-scale tuning phase. For discrete variables we kept the 80% most frequent values, and for continuous variables we kept the value ranging between the 10th and 90th percentiles.

For instance, on CTGAN the large-scale tuning included six encoder options: CDF, PLE_CDF, PTP, Quantile, MinMax, and CBN. But only CDF and PLE_CDF were selected on most datasets. We could also drastically reduce TabSyn VAE's learning rate range from $(10^{-5}, 10^{-2})$ to $(10^{-3}, 7 \cdot 10^{-3})$. We present all these reduced search spaces in Appendix A.

6.2. Putting the Reduced Search Spaces to the Test

To evaluate the reduced search spaces, we ran a new tuning on all datasets with only 50 trials and a strict limit of 20 minutes per trial. Since TabSyn's training is done in two steps, we allocated 10 minutes for the VAE training and 10 minutes for the denoiser. Each hyperparameter search was performed with the same 3-fold splits and the same methodology as in the extensive experiment.

6.2.1. A Cost-Performance Trade-off

In Figure 10 we summarize the performance of the models after limited-budget hyperparameter search against the ones obtained with the base models and the ones obtained after the extensive tuning of Section 4.3. On the x-axis we display the total GPU-time (search+optimization) needed to obtain the corresponding model and on the y-axis we show the C2ST performance. The diameters of the dots represents the number of possible configurations of the hyperparameter search space as described in Appendix A.

If we compare against the non-tuned base models, the C2ST performance of all models except TabSyn is clearly improved by the limited-budget hyperparameter tuning. For CTGAN, and TVAE the gain is huge: after only a few minutes of tuning with the reduced search space we reach the same performance as the one obtained after hours of heavy tuning. For TabDDPM, we also improved the base performance by running the limited-budget hyperparameter tuning. However, we did not reach the same performance as we did with an extensive search.

In the limited-budget experiment, the TabSyn's performance degradation against the base model reveals the importance of a well-trained VAE which leads to better samples from the diffusion process. Indeed, on Adult dataset for instance, within its 10 minutes budget, it was trained with less than 600

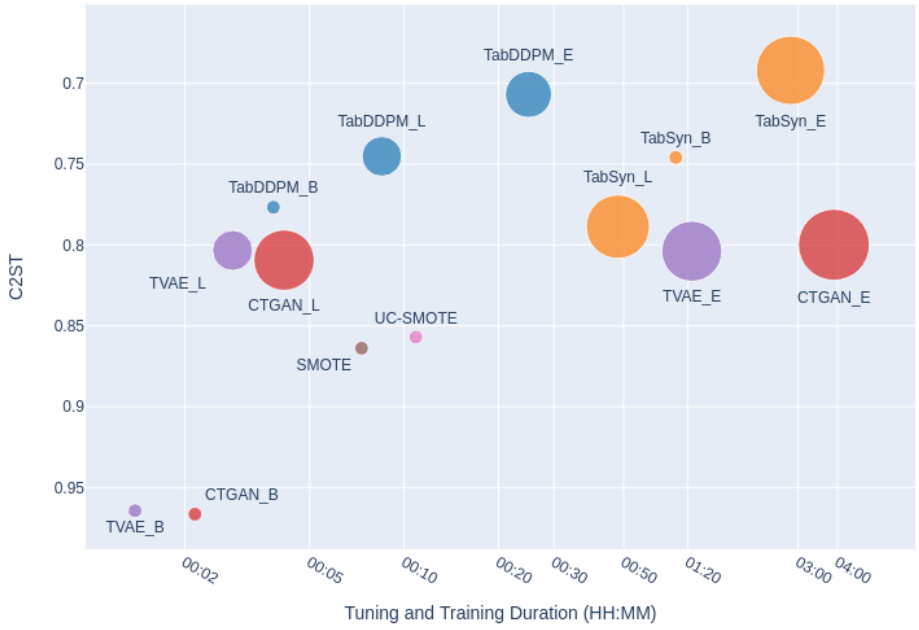


Figure 10: Average C2ST performance of the models under various setups (base models are appended with "_B", limited-budget with "_L" and extensive with "_E"). The C2ST axis shows the best models at the top, the dot diameter indicates the complexity of the search space. The duration axis and dot diameters are log-scaled for better visualization.

epochs against 4000 for the base model. This suggests to increase the number of epochs or the training time for TabSyn's VAE in further experiments.

Finally, as shown in Figure 10, the costs induced by the heavily optimized TabSyn model are high compared to the base one. The performance gain and the gap in cost should be considered depending on the task and constraints as the base model already delivers *decent* performance.

It is worth noting that, due to its quick implementation, TabDDPM performed sometimes more training steps within its 20 minutes time budget on the limited-budget experiment than it did on the extensive search where the number of steps was bounded. But on the other hand, its performance was also affected by the reduced number of trials (50 instead of 300).

6.2.2. Multi-criteria Analysis of the Limited-Budget Tuning

Model	Percentiles	C2ST ↓	DCR-R ↓	ML-EF ↑	Shape ↑	Pair ↑
Train Copy	75%	0.50	1.00	0.90	0.99	0.99
	50%	0.50	1.00	0.73	0.99	0.98
	25%	0.50	1.00	0.63	0.99	0.91
TVAE	75%	0.92	0.65	0.80	0.96	0.95
	50%	0.80	0.63	0.70	0.95	0.93
	25%	0.70	0.61	0.51	0.94	0.87
CTGAN	75%	0.89	0.64	0.71	0.98	0.96
	50%	0.85	0.63	0.68	0.97	0.92
	25%	0.70	0.62	0.45	0.95	0.85
TabDDPM	75%	0.83	0.67	0.83	0.97	0.95
	50%	0.72	0.63	0.69	0.96	0.91
	25%	0.63	0.62	0.50	0.95	0.81
TabSyn	75%	0.88	0.63	0.71	0.97	0.96
	50%	0.81	0.62	0.65	0.95	0.93
	25%	0.72	0.61	0.34	0.94	0.82

Table 5: Performance dispersion across all datasets in the ***limited-budget*** benchmark. DCR-R and ML-EF stand for DCR-Rate and ML-Efficacy respectively.

In Figure 11 we summarize the model's relative performances according to the five metrics for the limited-budget tuning experiment. To complete this point of view, Table 5 summarizes the quality metric distributions among all datasets and folds. The full results are detailed in Appendix F.

A first remark about Figure 11 is that it is more balanced than Figure 5: with a fair and limited GPU budget allocation, the models tends to perform similarly among all criteria.

There is no more clear leading model although TabDDPM obtains the best C2ST performance and TabSyn slightly dominates in terms of privacy (DCR-Rate). We however have a safe median DCR-Rate score of about 0.63 for this experiment on all neural models (against 0.62 for the extensive experiment). For ML-Efficacy, TabDDPM remains slightly the best but there is no clear leader either.

The reduced search space that we provide is hence a good starting point to perform a quick dataset-specific hyperparameters optimization that will fit on a medium-size workstation.

The relatively balanced results of this new experiment confirms that the superiority of a tabular data generator is not only due to its model but also to the whole tuning and training compute effort.

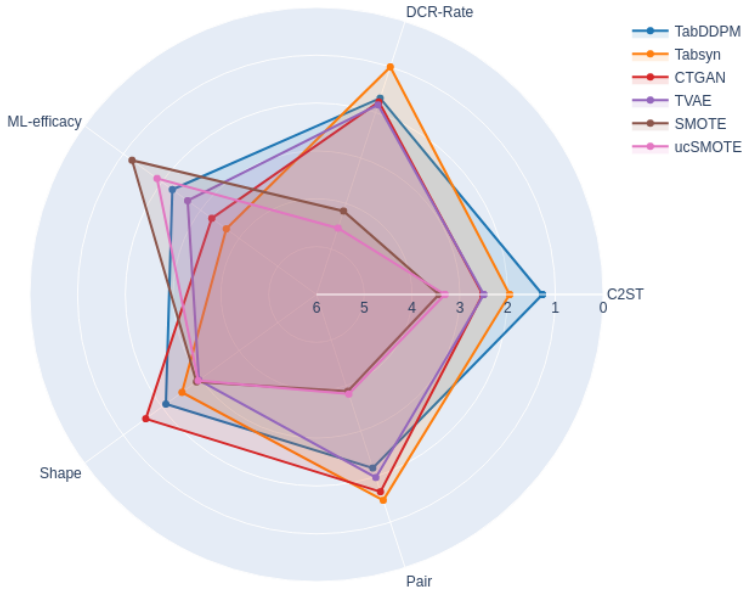


Figure 11: Radar chart of the optimized models in the limited-budget benchmark across various metrics. "Shape" stands for *column-wise similarity*, and "Pair" stands for *pair-wise correlation*.

Overall, TabDDPM provides a good balance between realism, privacy, utility, and cost. TVAE is a viable alternative if the utility constraint can be relaxed. It can be tuned using the limited-budget experiment search space to quickly achieve good performance. If resources are available and cost is not a priority, it is advisable to tune TabSyn, which achieves good realism results but at a higher cost. Finally, for quick results when privacy is not a concern and when the main focus is utility, SMOTE is the recommended approach.

7. Conclusion

We proposed a typology of state-of-art models for tabular data generation and we benchmarked five of them on 16 datasets with a strict 3-fold cross-validation procedure. The number of folds was limited to 3 because increasing it would have dramatically increased the costs and carbon footprint. We performed extensive large-scale tuning on a super computer from which we derived a reduced search-space. Moreover, we performed a limited-

budget benchmark that fits on a medium-size workstation. Leveraging these benchmarks we were able to provide several insights on the models while answering to three technical questions: (i) is it worth optimizing the hyperparameters/preprocessing specifically for each dataset? (ii) can we propose a reduced search space that fits well for all datasets? (iii) is there a clear trade-off between training/sampling costs, and synthetic data quality?

For the two first questions, Figure 10 is certainly the best summary: most models, including TabSyn, benefit greatly from a dataset-specific tuning. However, the whole tuning process is costly in terms of time, money, energy, and CO_2 emissions, and there is clearly a "diminishing return" effect. We conclude that, even for TabSyn, a quick dataset-specific model tuning based on the reduced search spaces we provide in Appendix A is enough to get most of the performance at the scale of a medium-size workstation. We also recommend using a subset of the dataset for hyperparameter tuning. The obtained hyperparameters can then be used to train the model at full-scale.

Regarding the trade-off question on the multiple considered criteria: realism, privacy, utility and costs, Figure 5 and Figure 11 confirm that if we do not limit the compute power, the two diffusion-based models TabDDPM and TabSyn are the most recommended solutions for tabular data generation. Nevertheless, the performance gaps between models become quite narrow with a limited compute power.

8. Future Work

Our study focuses on models trained, tuned, and evaluated on a single table and can be extended to evaluate cross-table models that fits with multiple table structures. This is an important step towards foundation models for tabular data generation. Models such as GReaT [38] offers an interesting approach for a cross-table approach by converting the data generation process into a text generation one. However, when evaluated on a single tables, it struggle against models like CTGAN [20] for joint probability distribution [7, 12]. Working on a cross-table foundation model also introduces additional considerations towards data poisoning or adversarial attacks as foundation models are proned to being poisoned [79].

Another important research direction is the models' interpretability. Understanding the generation procedure is also important to build user's trust for decision making. Such interpretability study can be performed starting from a tree-based model such as Forest-Flow [63].

9. Ackowlegment

This work was granted access to the HPC resources of IDRIS under the allocations 2023-AD011014381, 2023-AD011011407R3, and 2023-AD011012220R2 made by GENCI.

References

[1] D. Libes, D. Lechevalier, S. Jain, Issues in synthetic data generation for advanced manufacturing, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 1746–1754.

[2] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, G. Kasneci, Deep neural networks and tabular data: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022).

[3] L. Zhang, S. Zhang, K. Balog, Table2vec: Neural word and entity embeddings for table population and retrieval, in: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, 2019, pp. 1029–1032.

[4] S. Chitlangia, A. Muralidhar, R. Agarwal, Self supervised pre-training for large scale tabular data, in: NeurIPS 2022 Workshop on Table Representation Learning, 2022.

[5] T. Zhang, S. Wang, S. Yan, L. Jian, Q. Liu, Generative table pre-training empowers models for tabular prediction, in: H. Bouamor, J. Pino, K. Bali (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore, 2023, pp. 14836–14854. URL: https://aclanthology.org/2023.emnlp-main.917. doi:10.18653/v1/2023.emnlp-main.917.

[6] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data?, in: Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2022. URL: https://openreview.net/forum?id=Fp7__phQszn.

[7] H. Zhang, J. Zhang, Z. Shen, B. Srinivasan, X. Qin, C. Faloutsos, H. Rangwala, G. Karypis, Mixed-type tabular data synthesis with score-based diffusion in latent space, in: The Twelfth International Conference on Learning Representations, 2024.

[8] C. M. Bowen, J. Snoke, Comparative study of differentially private synthetic data algorithms from the nist pscr differential privacy synthetic data challenge, Journal of Privacy and Confidentiality 11 (2021). URL: https://journalprivacyconfidentiality.org/index.php/jpc/article/view/748. doi:10.29012/jpc.748.

[9] C. Arnold, M. Neunhoeffer, Really useful synthetic data–a framework to evaluate the quality of differentially private synthetic data, arXiv e-prints (2020) arXiv–2004.

[10] Y. Tao, R. McKenna, M. Hay, A. Machanavajjhala, G. Miklau, Benchmarking differentially private synthetic data generation algorithms (2022).

[11] Y. Hu, F. Wu, Q. Li, Y. Long, G. Garrido, C. Ge, B. Ding, D. Forsyth, B. Li, D. Song, Sok: Privacy-preserving data synthesis, in: 2024 IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, 2023, pp. 2–2.

[12] Y. Du, N. Li, Towards principled assessment of tabular data synthesis algorithms, arXiv preprint arXiv:2402.06806 (2024).

[13] G. Zabërgja, A. Kadra, J. Grabocka, Tabular data: Is attention all you need?, 2024. arXiv:2402.03970.

[14] J. Fonseca, F. Bacao, Tabular and latent space synthetic data generation: a literature review, Journal of Big Data 10 (2023) 115.

[15] A. Kotelnikov, D. Baranchuk, I. Rubachev, A. Babenko, Tabddpm: Modelling tabular data with diffusion models, in: International Conference on Machine Learning, PMLR, 2023, pp. 17564–17579.

[16] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, C. Faloutsos, Large language models on tabular data–a survey, arXiv preprint arXiv:2402.17944 (2024).

[17] D. P. Kingma, M. Welling, Auto-encoding variational bayes, stat 1050 (2014) 1.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Communications of the ACM 63 (2020) 139–144.

[19] E. G. Tabak, C. V. Turner, A family of nonparametric density estimation algorithms, Communications on Pure and Applied Mathematics 66 (2013) 145–164.

[20] L. Xu, M. Skoularidou, A. Cuesta-Infante, K. Veeramachaneni, Modeling tabular data using conditional gan, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems (NeurIPS), volume 32, Curran Associates, Inc., 2019.

[21] N. Patki, R. Wedge, K. Veeramachaneni, The synthetic data vault, in: IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2016, pp. 399–410. doi:10.1109/DSAA.2016.49.

[22] C. Ma, S. Tschiatschek, R. Turner, J. M. Hernández-Lobato, C. Zhang, Vaem: a deep generative model for heterogeneous mixed type data, Advances in Neural Information Processing Systems 33 (2020) 11237–11247.

[23] Z. Zhao, A. Kunar, R. Birke, L. Y. Chen, Ctab-gan: Effective table data synthesizing, in: Asian Conference on Machine Learning, PMLR, 2021, pp. 97–112.

[24] E. H. Zein, T. Urvoy, Tabular data generation: Can we fool XGBoost ?, in: NeurIPS 2022 First Table Representation Workshop, 2022.

[25] L. V. H. Vardhan, S. Kok, Generating privacy-preserving synthetic tabular data using oblivious variational autoencoders, in: Proceedings of the Workshop on Economics of Privacy and Data Labor at the 37 th International Conference on Machine Learning, 2020.

[26] T. Liu, Z. Qian, J. Berrevoets, M. van der Schaar, GOGGLE: Generative modelling for tabular data by learning relational structure, in: The

Eleventh International Conference on Learning Representations, 2023. URL: https://openreview.net/forum?id=fPVRcJqspu.

[27] J. Jordon, J. Yoon, M. van der Schaar, Pate-gan: Generating synthetic data with differential privacy guarantees, in: International Conference on Learning Representations, 2018. URL: https://api.semanticscholar.org/CorpusID:53342261.

[28] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, Y. Kim, Data synthesis based on generative adversarial networks, Proceedings of the VLDB Endowment 11 (2018).

[29] R. D. Camino, C. Hammerschmidt, et al., Generating multi-categorical samples with generative adversarial networks, in: ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models, 2018.

[30] M. K. Baowaly, C.-C. Lin, C.-L. Liu, K.-T. Chen, Synthesizing electronic health records using improved generative adversarial networks, Journal of the American Medical Informatics Association: JAMIA 26 (2019) 228–241.

[31] H. Chen, S. Jajodia, J. Liu, N. Park, V. Sokolov, V. Subrahmanian, Faketables: Using gans to generate functional dependency preserving tables with bounded real data, in: 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, International Joint Conferences on Artificial Intelligence, 2019, pp. 2074–2080.

[32] A. Koivu, M. Sairanen, A. Airola, T. Pahikkala, Synthetic minority oversampling of vital statistics data with generative adversarial networks, Journal of the American Medical Informatics Association 27 (2020) 1667–1674.

[33] R. Sauber-Cole, T. M. Khoshgoftaar, The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey, Journal of Big Data 9 (2022) 98.

[34] D. S. Watson, K. Blesch, J. Kapar, M. N. Wright, Adversarial random forests for density estimation and generative modeling, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2023, pp. 5357–5375.

[35] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: International conference on machine learning, PMLR, 2017, pp. 214–223.

[36] S. Bond-Taylor, A. Leach, Y. Long, C. G. Willcocks, Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 7327–7347. doi:10.1109/TPAMI.2021.3116668.

[37] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Gray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, in: A. H. Oh, A. Agarwal, D. Belgrave, K. Cho (Eds.), Advances in Neural Information Processing Systems, 2022. URL: https://openreview.net/forum?id=TG8KACxEON.

[38] V. Borisov, K. Sessler, T. Leemann, M. Pawelczyk, G. Kasneci, Language models are realistic tabular data generators, in: The Eleventh International Conference on Learning Representations (ICLR), 2023.

[39] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf, accessed: 2024-11-15.

[40] A. V. Solatorio, O. Dupriez, Realtabformer: Generating realistic relational and tabular data using transformers, arXiv preprint arXiv:2302.02041 (2023).

[41] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, in: International Conference on Machine Learning, PMLR, 2015, pp. 2256–2265.

[42] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Advances in neural information processing systems 33 (2020) 6840–6851.

[43] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, B. Poole, Score-based generative modeling through stochastic differential equations, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=PxTIG12RRHS.

[44] P. Dhariwal, A. Nichol, Diffusion models beat gans on image synthesis, Advances in neural information processing systems 34 (2021) 8780–8794.

[45] T. Karras, M. Aittala, T. Aila, S. Laine, Elucidating the design space of diffusion-based generative models, Advances in Neural Information Processing Systems 35 (2022) 26565–26577.

[46] G. Truda, Generating tabular datasets under differential privacy, arXiv preprint arXiv:2308.14784 (2023). TableDiffusion.

[47] J. Kim, C. Lee, N. Park, STaSy: Score-based tabular data synthesis (2022).

[48] C. Lee, J. Kim, N. Park, Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis, in: Proceedings of the 40th International Conference on Machine Learning, ICML'23, JMLR.org, 2023.

[49] A. Vahdat, K. Kreis, J. Kautz, Score-based generative modeling in latent space, Advances in neural information processing systems 34 (2021) 11287–11302.

[50] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10684–10695.

[51] R. B. Nelsen, An Introduction to Copulas, second ed., Springer, New York, NY, USA, 2006.

[52] Z. Li, Y. Zhao, J. Fu, Sync: A copula based framework for generating synthetic data from aggregated sources, in: 2020 International Conference on Data Mining Workshops (ICDMW), IEEE, 2020, pp. 571–578.

[53] S. Kamthe, S. Assefa, M. Deisenroth, Copula flows for synthetic data generation, arXiv preprint arXiv:2101.00598 (2021).

[54] D. Meyer, T. Nagler, Synthia: multidimensional synthetic data generation in python, Journal of Open Source Software 6 (2021) 2863. URL: https://doi.org/10.21105/joss.02863. doi:10.21105/joss.02863.

[55] D. Meyer, T. Nagler, R. J. Hogan, Copula-based synthetic data augmentation for machine-learning emulators, Geoscientific Model Development 14 (2021) 5205–5215. URL: https://doi.org/10.5194/gmd-14-5205-2021. doi:10.5194/gmd-14-5205-2021.

[56] R. Mckenna, D. Sheldon, G. Miklau, Graphical-model based estimation and inference for differential privacy, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 4435–4444. URL: https://proceedings.mlr.press/v97/mckenna19a.html.

[57] Z. Zhang, T. Wang, N. Li, J. Honorio, M. Backes, S. He, J. Chen, Y. Zhang, {PrivSyn}: Differentially private data synthesis, in: 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 929–946.

[58] J. Dahmen, D. J. Cook, Synsys: A synthetic data generation system for healthcare applications, Sensors (Basel, Switzerland) 19 (2019). URL: https://api.semanticscholar.org/CorpusID:75136844.

[59] B. Tang, H. He, Kerneladasyn: Kernel based adaptive synthetic data generation for imbalanced learning, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 664–671. doi:10.1109/CEC.2015.7256954.

[60] M. Baak, S. Brugman, I. Fridman Rojas, L. Dalmeida, R. E.Q. Urlus, J.-B. Oger, Synthsonic: Fast, probabilistic modeling and synthesis of tabular data, in: G. Camps-Valls, F. J. R. Ruiz, I. Valera (Eds.), Proceedings of The 25th International Conference on Artificial Intelligence and Statistics, volume 151 of *Proceedings of Machine Learning Research*, PMLR, 2022, pp. 4747–4763. URL: https://proceedings.mlr.press/v151/baak22a.html.

[61] R. McKenna, G. Miklau, D. Sheldon, Winning the nist contest: A scalable and general approach to differentially private synthetic data,

Journal of Privacy and Confidentiality 11 (2021). URL: https://journalprivacyconfidentiality.org/index.php/jpc/article/view/778. doi:10.29012/jpc.778.

[62] J. Jewson, L. Li, L. Battaglia, S. Hansen, D. Rossell, P. Zwiernik, Graphical model inference with external network data, cemmap working paper CWP20/22, London, 2022. URL: https://hdl.handle.net/10419/272832. doi:10.47004/wp.cem.2022.2022.

[63] A. Jolicoeur-Martineau, K. Fatras, T. Kachman, Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees, in: S. Dasgupta, S. Mandt, Y. Li (Eds.), Proceedings of The 27th International Conference on Artificial Intelligence and Statistics, volume 238 of *Proceedings of Machine Learning Research*, PMLR, 2024, pp. 1288–1296. URL: https://proceedings.mlr.press/v238/jolicoeur-martineau24a.html.

[64] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.

[65] L. Torgo, R. Ribeiro, B. Pfahringer, P. Branco, Smote for regression, volume 8154, 2013, pp. 378–389. doi:10.1007/978-3-642-40669-0_33.

[66] L. Camacho, G. Douzas, F. Bacao, Geometric smote for regression, Expert Systems with Applications 193 (2022) 1–8. doi:10.1016/j.eswa.2021.116387, camacho, L., Douzas, G., & Bacao, F. (2022). Geometric SMOTE for regression. Expert Systems with Applications, 193(May), 1-8. [116387]. https://doi.org/10.1016/j.eswa.2021.116387.

[67] D. Lopez-Paz, M. Oquab, Revisiting classifier two-sample tests, in: International Conference on Learning Representations, 2016.

[68] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[69] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, Advances in neural information processing systems 31 (2018).

[70] M. Platzer, T. Reutterer, Holdout-based empirical assessment of mixed-type synthetic data, Frontiers in big Data 4 (2021) 679939.

[71] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, I. Stoica, Tune: A research platform for distributed model selection and training, arXiv preprint arXiv:1807.05118 (2018).

[72] J. Bergstra, D. Yamins, D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: International conference on machine learning, PMLR, 2013, pp. 115–123.

[73] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 24, Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[74] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-tzur, M. Hardt, B. Recht, A. Talwalkar, A system for massively parallel hyperparameter tuning, in: I. Dhillon, D. Papailiopoulos, V. Sze (Eds.), Proceedings of Machine Learning and Systems, volume 2, 2020, pp. 230–246. URL: https://proceedings.mlsys.org/paper_files/paper/2020/file/a06f20b349c6cf09a6b171c71b88bbfc-Paper.pdf.

[75] Y. Gorishniy, I. Rubachev, A. Babenko, On embeddings for numerical features in tabular deep learning, Advances in Neural Information Processing Systems 35 (2022) 24991–25004.

[76] P. K. Dunn, G. K. Smyth, Randomized quantile residuals, Journal of Computational and graphical statistics 5 (1996) 236–244.

[77] P. Mettes, E. van der Pol, C. G. M. Snoek, Hyperspherical Prototype Networks, Curran Associates Inc., Red Hook, NY, USA, 2019.

[78] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine learning research 7 (2006) 1–30.

[79] A. Wan, E. Wallace, S. Shen, D. Klein, Poisoning language models during instruction tuning, in: Proceedings of the 40th International Conference on Machine Learning, ICML'23, JMLR.org, 2023.

Appendix A. Benchmark challengers Hyperparameters

Hyperparameters search space of TVAE is in Table A.6, for CTGAN in A.7, for TabSyn's VAE and MLP in A.8, for TabDDPM in A.9, for smote and ucsmote in A.10. These tables also present the reduced search spaces suggested and applied for the experiment of Section 6.

Parameter	**Possible values**	
	Extensive	**Reduced**
Learning rate	qLogUniform(1e-4, 1e-2, 1e-4)	qLogUniform(1e-4, 7.3e-03, 1e-4)
Batch size	[100, 500, 2000]	[100]
Embedding dim.	[16, 32, 64, 128, 256, 512]	[16, 32, 64]
Encoder dim.	[64, 128, 256, 512]	[256, 512]
Encoder depth	[2, 3, 4]	[2]
Decoder dim.	[64, 128, 256, 512]	[256, 512]
Decoder depth	[2, 3, 4]	[2, 4]
Loss factor	[3, 2, 1, 0.5]	[3, 2]
L2 scale	qLogUniform(1e-5, 1e-4, 1e-5)	qLogUniform(1e-5, 6.3e-5, 1e-6)
Numerical encoder	[CDF, PLE_CDF, PTP, QuantileTransformer, MinMaxScaler, CBN[11]]	[CDF]
Categorical encoder	[one-hot-encoder]	[one-hot-encoder]
Epochs	[400]	∞
Number of trials per fold	300	50

Table A.6: Hyperparameter search spaces of TVAE: extensive and limited-budget benchmarks.

[11] ClusterBasedNormalizer

Parameter	Possible values	
	Extensive	**Reduced**
Discriminator learning rate	qLogUniform(5e-5, 1e-2, 5e-5)	qLogUniform(4e-4, 2.1e-03, 5e-5)
Generator learning rate	qLogUniform(5e-5, 1e-2, 5e-5)	qLogUniform(5e-5, 1.3e-3, 5e-5)
Batch size	[50, 100, 250, 500, 1000]	[100, 500, 1000]
Embedding dim.	[32, 64, 128, 256]	[32, 128]
Generator dim.	[128, 256]	[128]
Generator depth	[2, 3, 4]	[3, 4]
Discriminator dim.	[128, 256]	[256]
Discriminator depth	[2, 3]	[2, 3]
Generator decay	qLogUniform(1e-6, 1e-5, 1e-6)	qLogUniform(1e-6, 6.4e-6, 1e-7)
Discriminator decay	qLogUniform(1e-6,1e-5, 1e-6)	qLogUniform(1e-6, 8e-6, 1e-6)
Log frequency	[False, True]	[False, True]
Numerical encoder	[CDF, PLE_CDF, PTP, QuantileTransformer, MinMaxScaler, CBN[11]]	[CDF, PLE_CDF]
Categorical encoder	[one-hot-encoder]	[one-hot-encoder]
Epochs	[400]	∞
Number of trials per fold	300	50

Table A.7: Hyperparameter search spaces of CTGAN for the extensive and limited-budget benchmarks.

Model	Parameter	Possible values	
		Extensive	**Reduced**
VAE	Learning rate	qLogUniform(5e-5, 1e-2, 5e-5)	qLogUniform(1.1e-3, 7.2e-3, 5e-5)
	Batch size	[1024, 2048, 4096]	[1024, 2048]
	Weight decay	qLogUniform(1e-6, 1e-5, 1e-6)	qLogUniform(1e-6, 5e-6, 1e-6)
	Token dim.	[2, 4]	[4]
	Number of head	[1, 2]	[1, 2]
	Factor	[8, 16, 32, 64]	[8, 16, 64]
	Number of layers	[1, 2, 3, 4]	[1, 2, 4]
	Max. beta	[1e-2]	[1e-2]
	Min. beta	[1e-5]	[1e-5]
	Lambda	[0.7, 0.8, 0.85, 0.9, 0.95]	[0.8, 0.85, 0.9, 0.95]
	Epochs	[4000]	∞
MLP	Learning Rate	qLogUniform(5e-5, 1e-2, 5e-5)	qLogUniform(7.7e-4, 2.5e-3, 1e-5)
	Weight Decay	qLogUniform(1e-6, 1e-5, 1e-6)	qLogUniform(1e-6, 3.3e-6, 1e-7)
	Batch size	[1024, 2048, 4096]	[1024, 4096]
	MLP's hidden dimension	[512, 1024]	[1024]
	Epochs	[2000]	∞
Number of trials per fold		100	50

Table A.8: Hyperparameter search spaces of TabSyn's VAE and MLP for the extensive and limited-budget benchmarks. In the second benchmark, the number of epochs was bounded by a time budget (10 minutes for the VAE, 10 minutes for the denoiser).

Parameter	Possible values	
	Extensive	**Reduced**
Batch size	[256, 4096]	[4096]
Dropout	[0.0]	[0.0]
Number of timesteps	[1000]	[1000]
Learning rate	qLogUniform(1e-5, 1e-3, 1e-5)	qLogUniform(3.5e-4, 9.2e-4, 1e-5)
Number of layers	[2, 4, 6, 8]	[2, 4, 6]
First layer's dim.	[128, 256, 512, 1024]	[256, 512, 1024]
Middle layer's dim.	[128, 256, 512, 1024]	[512, 1024]
Last layer's dim.	[128, 256, 512, 1024]	[256, 512, 1024]
Training iterations	[20000]	∞
Number of trials per fold	300	50

Table A.9: Hyperparameter search spaces of TabDDPM for the extensive and limited-budget benchmarks.

Parameter	Possible values
K-Neighbors	Grid search in range [2, 20]
Number of trials per fold	38

Table A.10: Hyperparameter search space of SMOTE and ucSMOTE.

Appendix B. Datasets Links

We provide the list of links toward the datasets that were used in this paper in Table B.11 below.

Domains	Dataset	URL
Environment	Abalone	www.openml.org/d/183
Social	Adult	www.openml.org/d/1590
Finance / Marketing	Bank	www.openml.org/d/1461
Marketing	Black Friday	www.openml.org/d/41540
Social / Environment	Bike Sharing	www.openml.org/d/42712
Environment	Covertype	www.openml.org/d/150
Health	Cardio	www.kaggle.com/sulianova/datasets
Finance / Marketing	Churn	www.kaggle.com/datasets/shrutimechlearn
Finance	Diamonds	www.openml.org/d/42225
Finance / Social	HELOC	www.kaggle.com/averkiyoliabev/datasets
Physics	Higgs	www.openml.org/d/4532
Finance / Social	House 16H	www.openml.org/d/574
Health / Insurance	Insurance	www.kaggle.com/datasets/mirichoi0218/insurance
Finance / Social	King	www.kaggle.com/datasets/harlfoxem/housesalesprediction
Physics	MiniBooNE	www.openml.org/d/41150
Synthetic	Moons	scikit-learn.org/stable/modules/classes.html

Table B.11: Domains and links to the datasets

Appendix C. Dataset-Level Results for Large-Scale Experiment

Table C.12 present the per-dataset performance according to the metrics described in Section 4.2 averaged on 3-folds with 5 samples per-fold.

Dataset	Model	Metrics						
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Train time ↓	Sample time ↓
Abalone	Train Copy	0.51 ± 0.00	1.00 ± 0.00	0.23 ± 0.01	0.96 ± 0.01	0.88 ± 0.01	-	-
	CTGAN	**0.73 ± 0.03**	**0.63 ± 0.02**	0.17 ± 0.01	0.93 ± 0.01	0.86 ± 0.03	685 ± 398.05	00 ± 0.01
	TVAE	0.75 ± 0.07	0.64 ± 0.05	0.23 ± 0.02	0.91 ± 0.02	0.86 ± 0.04	**113 ± 3.57**	**00 ± 0.00**
	TabDDPM	0.78 ± 0.01	0.67 ± 0.01	0.23 ± 0.01	0.94 ± 0.00	0.85 ± 0.03	169 ± 28.82	03 ± 0.62
	TabSyn	0.78 ± 0.01	0.63 ± 0.01	0.22 ± 0.01	0.94 ± 0.02	0.87 ± 0.00	1056 ± 183.20	00 ± 0.08
	SMOTE	1.00 ± 0.00	0.85 ± 0.02	0.50 ± 0.02	0.85 ± 0.01	0.72 ± 0.01	-	00 ± 0.01
	UC-SMOTE	0.89 ± 0.01	0.92 ± 0.02	**0.51 ± 0.03**	**0.95 ± 0.01**	**0.88 ± 0.03**	-	00 ± 0.00

Results per datasets and models under diverse metrics for the extensive search (continued).

Dataset	Model	Metrics						
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Train time ↓	Sample time ↓
Adult	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.71 ± 0.01	0.99 ± 0.00	0.98 ± 0.00	-	-
	CTGAN	0.74 ± 0.02	0.72 ± 0.00	0.67 ± 0.01	0.97 ± 0.00	0.88 ± 0.01	2755 ± 920.97	00 ± 0.01
	TVAE	0.77 ± 0.01	0.72 ± 0.00	0.63 ± 0.03	0.96 ± 0.00	0.91 ± 0.02	1404 ± 2.70	**00 ± 0.00**
	TabDDPM	0.65 ± 0.00	**0.62 ± 0.01**	0.67 ± 0.01	**0.98 ± 0.00**	0.95 ± 0.01	**443 ± 44.97**	10 ± 2.22
	TabSyn	**0.64 ± 0.01**	0.62 ± 0.00	0.66 ± 0.01	0.98 ± 0.00	**0.96 ± 0.01**	5042 ± 2119.47	01 ± 1.06
	SMOTE	0.93 ± 0.00	0.85 ± 0.00	**0.69 ± 0.01**	0.95 ± 0.00	0.90 ± 0.01	-	06 ± 0.11
	UC-SMOTE	0.93 ± 0.00	0.86 ± 0.01	0.67 ± 0.01	0.95 ± 0.00	0.90 ± 0.01	-	10 ± 0.03
Bank marketing	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.54 ± 0.01	0.99 ± 0.00	0.98 ± 0.01	-	-
	CTGAN	0.72 ± 0.01	0.63 ± 0.00	0.47 ± 0.02	0.98 ± 0.00	0.95 ± 0.01	3159 ± 65.25	00 ± 0.01
	TVAE	0.81 ± 0.04	0.63 ± 0.00	0.45 ± 0.08	0.96 ± 0.01	0.92 ± 0.02	1089 ± 628.11	**00 ± 0.02**
	TabDDPM	0.65 ± 0.01	**0.61 ± 0.00**	0.52 ± 0.01	**0.99 ± 0.01**	0.96 ± 0.01	**461 ± 42.63**	10 ± 1.98
	TabSyn	**0.61 ± 0.02**	0.62 ± 0.01	0.49 ± 0.02	0.99 ± 0.01	**0.97 ± 0.01**	2783 ± 471.56	02 ± 0.03
	SMOTE	0.79 ± 0.01	0.98 ± 0.00	**0.53 ± 0.02**	0.97 ± 0.00	0.95 ± 0.00	-	09 ± 0.16
	UC-SMOTE	0.79 ± 0.00	0.98 ± 0.00	0.46 ± 0.02	0.97 ± 0.00	0.95 ± 0.00	-	09 ± 0.03
Bike sharing	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.94 ± 0.00	0.99 ± 0.00	0.53 ± 0.00	-	-
	CTGAN	0.90 ± 0.01	**0.61 ± 0.00**	0.68 ± 0.02	**0.97 ± 0.01**	0.53 ± 0.00	3772 ± 2301.33	00 ± 0.05
	TVAE	0.81 ± 0.01	0.61 ± 0.00	0.78 ± 0.03	0.96 ± 0.01	0.53 ± 0.00	540 ± 15.46	**00 ± 0.00**
	TabDDPM	**0.81 ± 0.01**	0.62 ± 0.00	0.79 ± 0.01	0.97 ± 0.01	0.53 ± 0.00	**332 ± 13.26**	07 ± 0.90
	TabSyn	0.86 ± 0.02	0.62 ± 0.01	0.51 ± 0.07	0.96 ± 0.00	0.53 ± 0.00	1843 ± 556.82	00 ± 0.01
	SMOTE	0.98 ± 0.00	0.99 ± 0.01	0.85 ± 0.00	0.95 ± 0.00	**0.67 ± 0.08**	-	01 ± 0.06
	UC-SMOTE	0.98 ± 0.00	0.98 ± 0.00	**0.86 ± 0.01**	0.95 ± 0.01	0.67 ± 0.08	-	00 ± 0.09
Black friday	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.53 ± 0.01	1.00 ± 0.00	0.99 ± 0.00	-	-
	CTGAN	0.82 ± 0.02	0.66 ± 0.01	0.46 ± 0.02	0.97 ± 0.01	0.87 ± 0.00	10111 ± 8081.87	00 ± 0.12
	TVAE	0.87 ± 0.01	0.68 ± 0.01	0.42 ± 0.02	0.95 ± 0.01	0.91 ± 0.02	4978 ± 1.32	**00 ± 0.02**
	TabDDPM	0.87 ± 0.01	**0.64 ± 0.00**	0.47 ± 0.02	0.99 ± 0.01	**0.98 ± 0.01**	**366 ± 13.15**	39 ± 14.32
	TabSyn	0.87 ± 0.01	0.68 ± 0.00	0.17 ± 0.01	**0.99 ± 0.00**	0.46 ± 0.00	7533 ± 563.20	06 ± 3.54
	SMOTE	**0.80 ± 0.00**	0.97 ± 0.00	**0.50 ± 0.01**	0.95 ± 0.00	0.94 ± 0.00	-	31 ± 0.16
	UC-SMOTE	0.81 ± 0.01	0.97 ± 0.00	0.49 ± 0.01	0.94 ± 0.00	0.93 ± 0.00	-	106 ± 0.07
Cardio	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.72 ± 0.01	1.00 ± 0.00	0.98 ± 0.01	-	-
	CTGAN	0.62 ± 0.01	0.64 ± 0.00	0.70 ± 0.02	0.99 ± 0.01	0.96 ± 0.01	4678 ± 59.79	00 ± 0.01
	TVAE	0.72 ± 0.02	0.64 ± 0.01	0.72 ± 0.01	0.97 ± 0.01	0.95 ± 0.02	1708 ± 984.53	**00 ± 0.02**
	TabDDPM	**0.55 ± 0.01**	**0.62 ± 0.00**	0.72 ± 0.01	**0.99 ± 0.00**	**0.98 ± 0.01**	**185 ± 73.66**	13 ± 8.85
	TabSyn	0.56 ± 0.00	0.64 ± 0.00	0.72 ± 0.00	0.99 ± 0.00	0.98 ± 0.01	2960 ± 578.82	03 ± 0.01
	SMOTE	0.95 ± 0.00	0.98 ± 0.00	**0.73 ± 0.00**	0.93 ± 0.00	0.97 ± 0.01	-	00 ± 0.02
	UC-SMOTE	0.94 ± 0.00	0.98 ± 0.00	0.72 ± 0.01	0.93 ± 0.01	0.97 ± 0.01	-	01 ± 0.05
Churn	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.59 ± 0.02	0.95 ± 0.00	0.87 ± 0.01	-	-
	CTGAN	0.63 ± 0.02	0.64 ± 0.01	0.36 ± 0.03	0.93 ± 0.01	0.84 ± 0.02	2505 ± 953.49	00 ± 0.03
	TVAE	0.64 ± 0.01	0.63 ± 0.00	0.51 ± 0.00	0.92 ± 0.01	**0.84 ± 0.01**	**294 ± 6.20**	**00 ± 0.01**
	TabDDPM	0.66 ± 0.07	0.64 ± 0.05	0.50 ± 0.05	0.89 ± 0.06	0.82 ± 0.06	618 ± 35.47	25 ± 2.36
	TabSyn	**0.58 ± 0.02**	**0.61 ± 0.01**	**0.56 ± 0.03**	**0.93 ± 0.01**	0.48 ± 0.00	1627 ± 340.95	00 ± 0.22
	SMOTE	0.76 ± 0.01	0.86 ± 0.03	0.50 ± 0.02	0.87 ± 0.01	0.81 ± 0.02	-	01 ± 0.06
	UC-SMOTE	0.78 ± 0.02	0.93 ± 0.02	0.12 ± 0.09	0.87 ± 0.01	0.80 ± 0.02	-	00 ± 0.05

Results per datasets and models under diverse metrics for the extensive search (continued).

Dataset	Model	Metrics						
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Train time ↓	Sample time ↓
Covertype	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.90 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	-	-
	CTGAN	0.97 ± 0.01	0.60 ± 0.01	0.70 ± 0.01	0.98 ± 0.00	0.96 ± 0.01	51644 ± 17257.41	04 ± 0.60
	TVAE	0.90 ± 0.01	0.60 ± 0.00	0.77 ± 0.01	0.98 ± 0.00	0.96 ± 0.01	23219 ± 18238.02	**01 ± 0.44**
	TabDDPM	0.94 ± 0.01	**0.59 ± 0.04**	0.66 ± 0.04	0.95 ± 0.01	0.91 ± 0.01	**868 ± 30.61**	249 ± 25.67
	TabSyn	**0.63 ± 0.02**	0.62 ± 0.01	0.75 ± 0.02	0.99 ± 0.00	0.69 ± 0.00	4937 ± 1556.02	03 ± 0.05
	SMOTE	0.97 ± 0.00	0.97 ± 0.00	**0.90 ± 0.00**	**1.00 ± 0.00**	**1.00 ± 0.01**	-	125 ± 0.69
	UC-SMOTE	0.97 ± 0.00	0.97 ± 0.00	0.90 ± 0.00	1.00 ± 0.00	0.99 ± 0.01	-	138 ± 0.71
Diamonds	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.98 ± 0.00	0.99 ± 0.00	0.77 ± 0.01	-	-
	CTGAN	0.86 ± 0.01	0.65 ± 0.01	0.94 ± 0.01	0.97 ± 0.01	0.80 ± 0.04	3389 ± 2.57	00 ± 0.01
	TVAE	0.79 ± 0.03	0.64 ± 0.00	0.96 ± 0.01	0.94 ± 0.01	0.74 ± 0.04	1104 ± 637.85	**00 ± 0.02**
	TabDDPM	**0.71 ± 0.01**	**0.61 ± 0.00**	**0.97 ± 0.00**	**0.98 ± 0.01**	0.70 ± 0.02	**374 ± 13.27**	16 ± 6.28
	TabSyn	0.87 ± 0.01	0.65 ± 0.00	0.79 ± 0.10	0.98 ± 0.01	0.76 ± 0.04	3255 ± 360.89	02 ± 1.12
	SMOTE	0.97 ± 0.00	0.93 ± 0.01	0.92 ± 0.01	0.97 ± 0.00	0.77 ± 0.02	-	03 ± 0.02
	UC-SMOTE	0.97 ± 0.00	0.94 ± 0.01	0.93 ± 0.01	0.97 ± 0.00	**0.80 ± 0.04**	-	01 ± 0.04
Heloc	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.70 ± 0.01	0.99 ± 0.00	0.97 ± 0.01	-	-
	CTGAN	0.92 ± 0.01	**0.64 ± 0.01**	0.66 ± 0.05	**0.97 ± 0.01**	0.94 ± 0.02	4330 ± 3393.17	00 ± 0.10
	TVAE	0.87 ± 0.00	0.66 ± 0.01	0.70 ± 0.01	0.95 ± 0.00	0.94 ± 0.01	421 ± 249.85	**00 ± 0.00**
	TabDDPM	0.72 ± 0.01	0.64 ± 0.01	**0.70 ± 0.01**	0.97 ± 0.00	0.95 ± 0.01	**80 ± 9.69**	01 ± 0.53
	TabSyn	**0.70 ± 0.02**	0.66 ± 0.01	0.70 ± 0.02	0.97 ± 0.00	0.95 ± 0.01	1870 ± 302.36	00 ± 0.00
	SMOTE	0.92 ± 0.01	1.00 ± 0.00	0.69 ± 0.01	0.94 ± 0.00	**0.95 ± 0.01**	-	00 ± 0.04
	UC-SMOTE	0.92 ± 0.01	1.00 ± 0.00	0.69 ± 0.01	0.94 ± 0.00	0.95 ± 0.02	-	00 ± 0.02
Higgs	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.74 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	-	-
	CTGAN	0.85 ± 0.02	**0.60 ± 0.00**	0.70 ± 0.01	0.99 ± 0.01	0.98 ± 0.00	12696 ± 4197.49	00 ± 0.13
	TVAE	0.92 ± 0.01	0.60 ± 0.00	0.70 ± 0.02	0.94 ± 0.01	0.98 ± 0.01	5752 ± 116.50	**00 ± 0.08**
	TabDDPM	0.57 ± 0.00	0.63 ± 0.00	0.73 ± 0.00	**0.99 ± 0.00**	**0.99 ± 0.00**	**223 ± 10.78**	22 ± 2.25
	TabSyn	**0.57 ± 0.01**	0.61 ± 0.00	0.73 ± 0.00	0.94 ± 0.00	0.99 ± 0.00	7503 ± 2581.02	04 ± 2.17
	SMOTE	0.84 ± 0.00	1.00 ± 0.00	**0.73 ± 0.00**	0.96 ± 0.00	0.98 ± 0.01	-	00 ± 0.05
	UC-SMOTE	0.83 ± 0.00	1.00 ± 0.00	0.73 ± 0.00	0.96 ± 0.00	0.98 ± 0.01	-	01 ± 0.04
House 16h	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.64 ± 0.01	0.99 ± 0.01	0.99 ± 0.00	-	-
	CTGAN	0.84 ± 0.01	0.62 ± 0.00	0.47 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	983 ± 559.75	**00 ± 0.01**
	TVAE	0.82 ± 0.03	0.61 ± 0.01	0.48 ± 0.06	0.95 ± 0.01	0.98 ± 0.01	922 ± 6.68	00 ± 0.01
	TabDDPM	**0.60 ± 0.01**	**0.61 ± 0.00**	0.61 ± 0.02	**0.98 ± 0.00**	0.99 ± 0.01	**95 ± 21.99**	02 ± 0.88
	TabSyn	0.74 ± 0.01	0.62 ± 0.01	0.40 ± 0.02	0.97 ± 0.00	0.99 ± 0.00	2370 ± 810.03	01 ± 0.01
	SMOTE	0.87 ± 0.00	0.86 ± 0.05	**0.62 ± 0.01**	0.92 ± 0.01	**0.99 ± 0.00**	-	00 ± 0.03
	UC-SMOTE	0.87 ± 0.01	0.83 ± 0.05	0.62 ± 0.02	0.92 ± 0.01	0.99 ± 0.00	-	00 ± 0.02
Insurance	Train Copy	0.48 ± 0.01	1.00 ± 0.00	0.85 ± 0.03	0.96 ± 0.01	0.91 ± 0.01	-	-
	CTGAN	0.67 ± 0.03	0.92 ± 0.01	0.71 ± 0.01	**0.94 ± 0.00**	0.87 ± 0.02	265 ± 23.11	00 ± 0.00
	TVAE	0.67 ± 0.02	0.92 ± 0.01	0.77 ± 0.01	0.91 ± 0.01	0.86 ± 0.02	**32 ± 4.50**	**00 ± 0.00**
	TabDDPM	0.68 ± 0.02	0.62 ± 0.01	**0.83 ± 0.03**	0.93 ± 0.02	**0.89 ± 0.02**	181 ± 18.27	03 ± 1.01
	TabSyn	**0.60 ± 0.02**	**0.61 ± 0.02**	0.82 ± 0.00	0.94 ± 0.01	0.88 ± 0.00	1156 ± 74.54	00 ± 0.03
	SMOTE	0.68 ± 0.01	0.80 ± 0.03	0.81 ± 0.01	0.94 ± 0.01	0.87 ± 0.01	-	00 ± 0.01
	UC-SMOTE	0.67 ± 0.01	0.85 ± 0.02	0.80 ± 0.02	0.93 ± 0.02	0.87 ± 0.02	-	00 ± 0.00

Results per datasets and models under diverse metrics for the extensive search (continued).

Dataset	Model	Metrics						
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Train time ↓	Sample time ↓
King	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.87 ± 0.00	0.98 ± 0.01	0.97 ± 0.01	-	-
	CTGAN	**0.93 ± 0.01**	0.62 ± 0.00	0.72 ± 0.04	**0.97 ± 0.01**	0.94 ± 0.01	1805 ± 8.66	00 ± 0.01
	TVAE	0.94 ± 0.01	**0.61 ± 0.01**	0.81 ± 0.02	0.94 ± 0.01	0.94 ± 0.01	990 ± 6.97	**00 ± 0.00**
	TabDDPM	0.97 ± 0.01	0.73 ± 0.06	0.64 ± 0.20	0.71 ± 0.06	0.83 ± 0.01	**244 ± 14.03**	06 ± 0.61
	TabSyn	0.93 ± 0.01	0.62 ± 0.00	0.12 ± 0.03	0.96 ± 0.01	0.94 ± 0.00	2717 ± 648.93	01 ± 0.01
	SMOTE	0.98 ± 0.00	0.97 ± 0.00	**0.83 ± 0.01**	0.92 ± 0.01	0.95 ± 0.01	-	02 ± 0.23
	UC-SMOTE	0.98 ± 0.00	0.96 ± 0.01	0.82 ± 0.03	0.92 ± 0.01	**0.96 ± 0.01**	-	00 ± 0.03
Miniboone	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.89 ± 0.01	0.99 ± 0.00	0.99 ± 0.01	-	-
	CTGAN	0.91 ± 0.02	0.60 ± 0.01	0.83 ± 0.03	0.96 ± 0.02	0.58 ± 0.01	6879 ± 2388.04	00 ± 0.06
	TVAE	0.95 ± 0.03	**0.60 ± 0.00**	0.86 ± 0.00	0.93 ± 0.01	0.59 ± 0.01	4876 ± 4730.59	**00 ± 0.05**
	TabDDPM	0.64 ± 0.02	0.61 ± 0.01	0.89 ± 0.00	**0.99 ± 0.00**	0.91 ± 0.03	**193 ± 46.01**	23 ± 8.73
	TabSyn	**0.61 ± 0.01**	0.61 ± 0.00	0.89 ± 0.00	0.99 ± 0.01	0.91 ± 0.03	7861 ± 1700.48	07 ± 0.08
	SMOTE	0.83 ± 0.00	1.00 ± 0.00	**0.89 ± 0.00**	0.97 ± 0.01	**0.99 ± 0.01**	-	01 ± 0.10
	UC-SMOTE	0.83 ± 0.00	1.00 ± 0.00	0.89 ± 0.00	0.97 ± 0.01	0.98 ± 0.01	-	03 ± 0.04
Moons	Train Copy	0.50 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	-	-
	CTGAN	0.68 ± 0.01	0.61 ± 0.01	**1.00 ± 0.00**	0.96 ± 0.01	0.97 ± 0.01	2571 ± 967.05	00 ± 0.02
	TVAE	0.63 ± 0.01	0.61 ± 0.02	1.00 ± 0.00	0.98 ± 0.00	0.97 ± 0.00	752 ± 51.10	**00 ± 0.01**
	TabDDPM	**0.52 ± 0.01**	0.61 ± 0.01	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	**183 ± 92.44**	09 ± 6.74
	TabSyn	0.54 ± 0.01	**0.61 ± 0.01**	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.01	2647 ± 656.79	01 ± 0.81
	SMOTE	0.54 ± 0.00	0.84 ± 0.05	1.00 ± 0.00	**0.99 ± 0.00**	**0.99 ± 0.00**	-	00 ± 0.02
	UC-SMOTE	0.54 ± 0.01	0.80 ± 0.03	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.01	-	00 ± 0.03

Table C.12: Results per dataset and model under diverse metrics. The training times (gradient descent) and sampling times are given in seconds. The sampling times are given for 5 samples. The best values per metric are formatted in **bold green** and the worse values are in red.

Appendix D. Training and Sampling Impact

We performed large-scale experiments including a quite costly hyperparameter tuning step. Note that the term "*costs*" in this section refers to the duration, energy consumption, and CO_2 emissions. We evaluate the training and sampling costs of the models with their optimized hyperparameters, as well as the full hyperparameter tuning costs.

Appendix D.1. Training and Sampling Costs

Due to the massive nature of the experiments, the hyperparameter search could not all be run on the same hardware. We hence estimated the training (gradient descent) cost by running all models on a single Tesla V100-SXM2 32 GB. Table D.13 provides the raw training and sampling energy consumption

and emissions for reference. All of the TabSyn costs shown are a combination of the VAE cost and the denoiser cost, which are estimated separately and then added together. Also, all TabSyn costs on the *Covertype* dataset are estimated considering the complete dataset size.

Dataset	Model	Emissions Training ↓	Energy Training ↓	Emissions Sampling ↓	Energy Sampling ↓
Abalone	CTGAN	$3.05 \pm 1.77 \cdot 10^{-3}$	$4.53 \pm 2.62 \cdot 10^{-2}$	$9.74 \pm 2.37 \cdot 10^{-8}$	$1.45 \pm 0.35 \cdot 10^{-6}$
	TabDDPM	$\mathbf{9.16 \pm 1.42 \cdot 10^{-6}}$	$\mathbf{1.36 \pm 0.21 \cdot 10^{-4}}$	$2.90 \pm 0.50 \cdot 10^{-5}$	$4.31 \pm 0.74 \cdot 10^{-4}$
	TabSyn	$3.18 \pm 0.32 \cdot 10^{-4}$	$4.27 \pm 0.32 \cdot 10^{-3}$	$1.49 \pm 0.78 \cdot 10^{-6}$	$2.21 \pm 1.16 \cdot 10^{-5}$
	TVAE	$5.06 \pm 0.13 \cdot 10^{-4}$	$7.51 \pm 0.19 \cdot 10^{-3}$	$\mathbf{6.23 \pm 2.14 \cdot 10^{-8}}$	$\mathbf{9.24 \pm 3.17 \cdot 10^{-7}}$
	SMOTE	-	-	$4.04 \pm 0.35 \cdot 10^{-7}$	$5.99 \pm 0.51 \cdot 10^{-6}$
	UC-SMOTE	-	-	$1.37 \pm 0.07 \cdot 10^{-7}$	$2.03 \pm 0.11 \cdot 10^{-6}$
Adult	CTGAN	$1.24 \pm 0.41 \cdot 10^{-2}$	$1.84 \pm 0.61 \cdot 10^{-1}$	$9.67 \pm 0.41 \cdot 10^{-7}$	$1.43 \pm 0.06 \cdot 10^{-5}$
	TabDDPM	$\mathbf{2.68 \pm 0.35 \cdot 10^{-4}}$	$\mathbf{3.98 \pm 0.51 \cdot 10^{-3}}$	$8.41 \pm 2.01 \cdot 10^{-5}$	$1.25 \pm 0.30 \cdot 10^{-3}$
	TabSyn	$1.68 \pm 0.72 \cdot 10^{-3}$	$1.42 \pm 0.36 \cdot 10^{-2}$	$1.56 \pm 0.85 \cdot 10^{-5}$	$2.31 \pm 1.26 \cdot 10^{-4}$
	TVAE	$6.30 \pm 0.01 \cdot 10^{-3}$	$9.34 \pm 0.01 \cdot 10^{-2}$	$\mathbf{3.54 \pm 0.07 \cdot 10^{-7}}$	$\mathbf{5.25 \pm 0.11 \cdot 10^{-6}}$
	SMOTE	-	-	$2.46 \pm 0.04 \cdot 10^{-5}$	$3.66 \pm 0.06 \cdot 10^{-4}$
	UC-SMOTE	-	-	$3.82 \pm 0.01 \cdot 10^{-5}$	$5.67 \pm 0.01 \cdot 10^{-4}$
Bank marketing	CTGAN	$1.41 \pm 0.03 \cdot 10^{-2}$	$2.09 \pm 0.04 \cdot 10^{-1}$	$8.88 \pm 0.32 \cdot 10^{-7}$	$1.32 \pm 0.05 \cdot 10^{-5}$
	TabDDPM	$\mathbf{2.50 \pm 0.22 \cdot 10^{-4}}$	$\mathbf{3.71 \pm 0.33 \cdot 10^{-3}}$	$7.81 \pm 1.91 \cdot 10^{-5}$	$1.16 \pm 0.28 \cdot 10^{-3}$
	TabSyn	$1.06 \pm 0.08 \cdot 10^{-3}$	$1.46 \pm 0.04 \cdot 10^{-2}$	$1.80 \pm 0.02 \cdot 10^{-5}$	$2.67 \pm 0.03 \cdot 10^{-4}$
	TVAE	$4.90 \pm 2.83 \cdot 10^{-3}$	$7.27 \pm 4.19 \cdot 10^{-2}$	$\mathbf{2.84 \pm 0.47 \cdot 10^{-7}}$	$\mathbf{4.22 \pm 0.69 \cdot 10^{-6}}$
	SMOTE	-	-	$3.46 \pm 0.05 \cdot 10^{-5}$	$5.13 \pm 0.08 \cdot 10^{-4}$
	UC-SMOTE	-	-	$3.32 \pm 0.01 \cdot 10^{-5}$	$4.92 \pm 0.02 \cdot 10^{-4}$
Bike sharing	CTGAN	$1.68 \pm 1.03 \cdot 10^{-2}$	$2.50 \pm 1.52 \cdot 10^{-1}$	$5.79 \pm 2.48 \cdot 10^{-7}$	$8.59 \pm 3.68 \cdot 10^{-6}$
	TabDDPM	$\mathbf{8.73 \pm 0.70 \cdot 10^{-5}}$	$\mathbf{1.30 \pm 0.10 \cdot 10^{-3}}$	$6.04 \pm 0.72 \cdot 10^{-5}$	$8.96 \pm 1.07 \cdot 10^{-4}$
	TabSyn	$6.29 \pm 1.72 \cdot 10^{-4}$	$8.11 \pm 0.27 \cdot 10^{-3}$	$7.02 \pm 0.03 \cdot 10^{-6}$	$1.04 \pm 0.00 \cdot 10^{-4}$
	TVAE	$2.42 \pm 0.07 \cdot 10^{-3}$	$3.59 \pm 0.11 \cdot 10^{-2}$	$\mathbf{1.37 \pm 0.27 \cdot 10^{-7}}$	$\mathbf{2.03 \pm 0.40 \cdot 10^{-6}}$
	SMOTE	-	-	$4.53 \pm 0.22 \cdot 10^{-6}$	$6.72 \pm 0.33 \cdot 10^{-5}$
	UC-SMOTE	-	-	$1.48 \pm 0.32 \cdot 10^{-6}$	$2.20 \pm 0.48 \cdot 10^{-5}$
Black friday	CTGAN	$4.55 \pm 3.60 \cdot 10^{-2}$	$6.75 \pm 5.34 \cdot 10^{-1}$	$2.70 \pm 0.56 \cdot 10^{-6}$	$4.01 \pm 0.83 \cdot 10^{-5}$
	TabDDPM	$\mathbf{8.97 \pm 1.75 \cdot 10^{-4}}$	$\mathbf{1.33 \pm 0.26 \cdot 10^{-2}}$	$2.98 \pm 1.09 \cdot 10^{-4}$	$4.42 \pm 1.62 \cdot 10^{-3}$
	TabSyn	$2.78 \pm 0.26 \cdot 10^{-3}$	$2.58 \pm 0.90 \cdot 10^{-2}$	$5.18 \pm 2.81 \cdot 10^{-5}$	$7.69 \pm 4.17 \cdot 10^{-4}$
	TVAE	$2.23 \pm 0.01 \cdot 10^{-2}$	$3.31 \pm 0.01 \cdot 10^{-1}$	$\mathbf{8.59 \pm 0.65 \cdot 10^{-7}}$	$\mathbf{1.27 \pm 0.10 \cdot 10^{-5}}$
	SMOTE	-	-	$1.14 \pm 0.01 \cdot 10^{-4}$	$1.68 \pm 0.01 \cdot 10^{-3}$
	UC-SMOTE	-	-	$3.84 \pm 0.02 \cdot 10^{-4}$	$5.70 \pm 0.03 \cdot 10^{-3}$
Cardio	CTGAN	$2.11 \pm 0.03 \cdot 10^{-2}$	$3.13 \pm 0.04 \cdot 10^{-1}$	$1.30 \pm 0.03 \cdot 10^{-6}$	$1.93 \pm 0.04 \cdot 10^{-5}$
	TabDDPM	$\mathbf{2.22 \pm 1.13 \cdot 10^{-4}}$	$\mathbf{3.30 \pm 1.68 \cdot 10^{-3}}$	$1.05 \pm 0.70 \cdot 10^{-4}$	$1.56 \pm 1.04 \cdot 10^{-3}$
	TabSyn	$1.15 \pm 0.16 \cdot 10^{-3}$	$1.67 \pm 0.03 \cdot 10^{-2}$	$2.89 \pm 0.01 \cdot 10^{-5}$	$4.29 \pm 0.02 \cdot 10^{-4}$
	TVAE	$7.64 \pm 4.39 \cdot 10^{-3}$	$1.13 \pm 0.65 \cdot 10^{-1}$	$\mathbf{2.70 \pm 0.94 \cdot 10^{-7}}$	$\mathbf{4.00 \pm 1.40 \cdot 10^{-6}}$
	SMOTE	-	-	$2.27 \pm 0.05 \cdot 10^{-6}$	$3.37 \pm 0.07 \cdot 10^{-5}$
	UC-SMOTE	-	-	$4.27 \pm 0.16 \cdot 10^{-6}$	$6.34 \pm 0.24 \cdot 10^{-5}$
Churn	CTGAN	$1.15 \pm 0.43 \cdot 10^{-2}$	$1.71 \pm 0.63 \cdot 10^{-1}$	$6.61 \pm 1.15 \cdot 10^{-7}$	$9.81 \pm 1.71 \cdot 10^{-6}$
	TabDDPM	$\mathbf{1.10 \pm 0.07 \cdot 10^{-4}}$	$\mathbf{1.63 \pm 0.10 \cdot 10^{-3}}$	$1.96 \pm 0.17 \cdot 10^{-4}$	$2.90 \pm 0.25 \cdot 10^{-3}$
	TabSyn	$4.97 \pm 0.79 \cdot 10^{-4}$	$6.25 \pm 0.62 \cdot 10^{-3}$	$3.43 \pm 1.81 \cdot 10^{-6}$	$5.09 \pm 2.68 \cdot 10^{-5}$
	TVAE	$1.34 \pm 0.03 \cdot 10^{-3}$	$1.98 \pm 0.05 \cdot 10^{-2}$	$\mathbf{3.08 \pm 0.29 \cdot 10^{-7}}$	$\mathbf{4.57 \pm 0.42 \cdot 10^{-6}}$
	SMOTE	-	-	$4.13 \pm 0.20 \cdot 10^{-6}$	$6.12 \pm 0.29 \cdot 10^{-5}$
	UC-SMOTE	-	-	$3.53 \pm 0.19 \cdot 10^{-6}$	$5.24 \pm 0.28 \cdot 10^{-5}$

Tuning costs estimation for the best models (continued).

Dataset	Model	Emissions Training ↓	Energy Training ↓	Emissions Sampling ↓	Energy Sampling ↓
Covertype	CTGAN	$2.33 \pm 0.78 \cdot 10^{-1}$	3.46 ± 1.15	$1.92 \pm 0.27 \cdot 10^{-5}$	$2.85 \pm 0.41 \cdot 10^{-4}$
	TabDDPM	$6.22 \pm 0.35 \cdot 10^{-3}$	$9.23 \pm 0.53 \cdot 10^{-2}$	$1.80 \pm 0.26 \cdot 10^{-3}$	$2.67 \pm 0.38 \cdot 10^{-2}$
	TabSyn	$\mathbf{1.82 \pm 0.68 \cdot 10^{-3}}$	$\mathbf{1.26 \pm 0.72 \cdot 10^{-2}}$	$2.37 \pm 0.04 \cdot 10^{-5}$	$3.52 \pm 0.06 \cdot 10^{-4}$
	TVAE	$1.04 \pm 0.81 \cdot 10^{-1}$	1.54 ± 1.21	$\mathbf{8.00 \pm 1.93 \cdot 10^{-6}}$	$\mathbf{1.19 \pm 0.29 \cdot 10^{-4}}$
	SMOTE	-	-	$4.52 \pm 0.02 \cdot 10^{-4}$	$6.71 \pm 0.04 \cdot 10^{-3}$
	UC-SMOTE	-	-	$5.01 \pm 0.03 \cdot 10^{-4}$	$7.43 \pm 0.04 \cdot 10^{-3}$
Diamonds	CTGAN	$1.52 \pm 0.00 \cdot 10^{-2}$	$2.26 \pm 0.01 \cdot 10^{-1}$	$8.44 \pm 0.13 \cdot 10^{-7}$	$1.25 \pm 0.02 \cdot 10^{-5}$
	TabDDPM	$\mathbf{3.33 \pm 0.71 \cdot 10^{-4}}$	$\mathbf{4.93 \pm 1.06 \cdot 10^{-3}}$	$1.30 \pm 0.47 \cdot 10^{-4}$	$1.92 \pm 0.70 \cdot 10^{-3}$
	TabSyn	$1.21 \pm 0.18 \cdot 10^{-3}$	$1.36 \pm 0.49 \cdot 10^{-2}$	$1.67 \pm 0.89 \cdot 10^{-5}$	$2.48 \pm 1.32 \cdot 10^{-4}$
	TVAE	$4.93 \pm 2.84 \cdot 10^{-3}$	$7.31 \pm 4.21 \cdot 10^{-2}$	$\mathbf{2.42 \pm 0.76 \cdot 10^{-7}}$	$\mathbf{3.59 \pm 1.12 \cdot 10^{-6}}$
	SMOTE	-	-	$1.32 \pm 0.01 \cdot 10^{-5}$	$1.96 \pm 0.01 \cdot 10^{-4}$
	UC-SMOTE	-	-	$3.67 \pm 0.14 \cdot 10^{-6}$	$5.44 \pm 0.21 \cdot 10^{-5}$
Heloc	CTGAN	$1.93 \pm 1.51 \cdot 10^{-2}$	$2.87 \pm 2.24 \cdot 10^{-1}$	$6.03 \pm 4.19 \cdot 10^{-7}$	$8.95 \pm 6.22 \cdot 10^{-6}$
	TabDDPM	$\mathbf{1.21 \pm 0.29 \cdot 10^{-5}}$	$\mathbf{1.80 \pm 0.43 \cdot 10^{-4}}$	$1.50 \pm 0.42 \cdot 10^{-5}$	$2.23 \pm 0.63 \cdot 10^{-4}$
	TabSyn	$5.78 \pm 0.66 \cdot 10^{-4}$	$6.91 \pm 0.18 \cdot 10^{-3}$	$4.60 \pm 0.11 \cdot 10^{-6}$	$6.83 \pm 0.16 \cdot 10^{-5}$
	TVAE	$1.88 \pm 1.11 \cdot 10^{-3}$	$2.80 \pm 1.65 \cdot 10^{-2}$	$\mathbf{1.15 \pm 0.43 \cdot 10^{-7}}$	$\mathbf{1.71 \pm 0.63 \cdot 10^{-6}}$
	SMOTE	-	-	$9.32 \pm 1.28 \cdot 10^{-7}$	$1.38 \pm 0.19 \cdot 10^{-5}$
	UC-SMOTE	-	-	$2.02 \pm 0.07 \cdot 10^{-6}$	$3.00 \pm 0.10 \cdot 10^{-5}$
Higgs	CTGAN	$5.71 \pm 1.88 \cdot 10^{-2}$	$8.47 \pm 2.79 \cdot 10^{-1}$	$2.92 \pm 0.56 \cdot 10^{-6}$	$4.33 \pm 0.84 \cdot 10^{-5}$
	TabDDPM	$\mathbf{4.01 \pm 0.25 \cdot 10^{-4}}$	$\mathbf{5.96 \pm 0.37 \cdot 10^{-3}}$	$1.77 \pm 0.18 \cdot 10^{-4}$	$2.63 \pm 0.27 \cdot 10^{-3}$
	TabSyn	$2.88 \pm 1.12 \cdot 10^{-3}$	$2.07 \pm 0.74 \cdot 10^{-2}$	$3.20 \pm 1.72 \cdot 10^{-5}$	$4.75 \pm 2.55 \cdot 10^{-4}$
	TVAE	$2.57 \pm 0.05 \cdot 10^{-2}$	$3.81 \pm 0.08 \cdot 10^{-1}$	$\mathbf{1.06 \pm 0.34 \cdot 10^{-6}}$	$\mathbf{1.58 \pm 0.51 \cdot 10^{-5}}$
	SMOTE	-	-	$2.01 \pm 0.19 \cdot 10^{-6}$	$2.98 \pm 0.28 \cdot 10^{-5}$
	UC-SMOTE	-	-	$6.19 \pm 0.13 \cdot 10^{-6}$	$9.18 \pm 0.20 \cdot 10^{-5}$
House 16h	CTGAN	$4.42 \pm 2.48 \cdot 10^{-3}$	$6.55 \pm 3.68 \cdot 10^{-2}$	$\mathbf{1.24 \pm 0.16 \cdot 10^{-7}}$	$\mathbf{1.85 \pm 0.24 \cdot 10^{-6}}$
	TabDDPM	$\mathbf{3.81 \pm 1.14 \cdot 10^{-5}}$	$\mathbf{5.66 \pm 1.69 \cdot 10^{-4}}$	$2.03 \pm 0.70 \cdot 10^{-5}$	$3.01 \pm 1.04 \cdot 10^{-4}$
	TabSyn	$8.56 \pm 2.73 \cdot 10^{-4}$	$9.40 \pm 1.12 \cdot 10^{-3}$	$9.68 \pm 0.08 \cdot 10^{-6}$	$1.44 \pm 0.01 \cdot 10^{-4}$
	TVAE	$4.12 \pm 0.02 \cdot 10^{-3}$	$6.11 \pm 0.03 \cdot 10^{-2}$	$1.40 \pm 0.17 \cdot 10^{-7}$	$2.08 \pm 0.26 \cdot 10^{-6}$
	SMOTE	-	-	$2.92 \pm 1.19 \cdot 10^{-7}$	$4.34 \pm 1.77 \cdot 10^{-6}$
	UC-SMOTE	-	-	$4.04 \pm 0.48 \cdot 10^{-7}$	$5.99 \pm 0.72 \cdot 10^{-6}$
Insurance	CTGAN	$1.18 \pm 0.10 \cdot 10^{-3}$	$1.75 \pm 0.15 \cdot 10^{-2}$	$7.40 \pm 1.62 \cdot 10^{-8}$	$1.10 \pm 0.24 \cdot 10^{-6}$
	TabDDPM	$\mathbf{2.83 \pm 0.29 \cdot 10^{-6}}$	$\mathbf{4.20 \pm 0.43 \cdot 10^{-5}}$	$2.65 \pm 0.94 \cdot 10^{-5}$	$3.93 \pm 1.40 \cdot 10^{-4}$
	TabSyn	$3.45 \pm 0.04 \cdot 10^{-4}$	$4.64 \pm 0.36 \cdot 10^{-3}$	$5.87 \pm 2.03 \cdot 10^{-7}$	$8.71 \pm 3.01 \cdot 10^{-6}$
	TVAE	$1.44 \pm 0.20 \cdot 10^{-4}$	$2.14 \pm 0.29 \cdot 10^{-3}$	$\mathbf{4.70 \pm 1.02 \cdot 10^{-8}}$	$\mathbf{6.98 \pm 1.51 \cdot 10^{-7}}$
	SMOTE	-	-	$1.83 \pm 0.26 \cdot 10^{-7}$	$2.71 \pm 0.39 \cdot 10^{-6}$
	UC-SMOTE	-	-	$1.02 \pm 0.04 \cdot 10^{-7}$	$1.51 \pm 0.06 \cdot 10^{-6}$
King	CTGAN	$8.15 \pm 0.04 \cdot 10^{-3}$	$1.21 \pm 0.01 \cdot 10^{-1}$	$5.72 \pm 0.34 \cdot 10^{-7}$	$8.49 \pm 0.50 \cdot 10^{-6}$
	TabDDPM	$\mathbf{9.27 \pm 0.56 \cdot 10^{-5}}$	$\mathbf{1.38 \pm 0.08 \cdot 10^{-3}}$	$5.08 \pm 0.49 \cdot 10^{-5}$	$7.55 \pm 0.73 \cdot 10^{-4}$
	TabSyn	$9.40 \pm 1.26 \cdot 10^{-4}$	$9.57 \pm 0.35 \cdot 10^{-3}$	$9.15 \pm 0.02 \cdot 10^{-6}$	$1.36 \pm 0.00 \cdot 10^{-4}$
	TVAE	$4.43 \pm 0.04 \cdot 10^{-3}$	$6.57 \pm 0.07 \cdot 10^{-2}$	$\mathbf{2.59 \pm 0.18 \cdot 10^{-7}}$	$\mathbf{3.85 \pm 0.26 \cdot 10^{-6}}$
	SMOTE	-	-	$8.77 \pm 0.84 \cdot 10^{-6}$	$1.30 \pm 0.12 \cdot 10^{-4}$
	UC-SMOTE	-	-	$2.25 \pm 0.10 \cdot 10^{-6}$	$3.34 \pm 0.14 \cdot 10^{-5}$

Tuning costs estimation for the best models (continued).

Dataset	Model	Emissions Training ↓	Energy Training ↓	Emissions Sampling ↓	Energy Sampling ↓
Miniboone	CTGAN	$3.16 \pm 1.09 \cdot 10^{-2}$	$4.69 \pm 1.62 \cdot 10^{-1}$	$3.50 \pm 0.24 \cdot 10^{-6}$	$5.20 \pm 0.35 \cdot 10^{-5}$
	TabDDPM	**$4.50 \pm 1.33 \cdot 10^{-4}$**	**$6.68 \pm 1.97 \cdot 10^{-3}$**	$1.84 \pm 0.69 \cdot 10^{-4}$	$2.74 \pm 1.03 \cdot 10^{-3}$
	TabSyn	$3.41 \pm 0.61 \cdot 10^{-3}$	$3.18 \pm 0.79 \cdot 10^{-2}$	$5.57 \pm 0.07 \cdot 10^{-5}$	$8.26 \pm 0.11 \cdot 10^{-4}$
	TVAE	$2.18 \pm 2.12 \cdot 10^{-2}$	$3.24 \pm 3.14 \cdot 10^{-1}$	**$1.94 \pm 0.22 \cdot 10^{-6}$**	**$2.88 \pm 0.32 \cdot 10^{-5}$**
	SMOTE	-	-	$4.39 \pm 0.34 \cdot 10^{-6}$	$6.51 \pm 0.51 \cdot 10^{-5}$
	UC-SMOTE	-	-	$1.32 \pm 0.01 \cdot 10^{-5}$	$1.95 \pm 0.02 \cdot 10^{-4}$
Moons	CTGAN	$1.15 \pm 0.43 \cdot 10^{-2}$	$1.71 \pm 0.64 \cdot 10^{-1}$	$5.02 \pm 1.07 \cdot 10^{-7}$	$7.45 \pm 1.59 \cdot 10^{-6}$
	TabDDPM	**$1.31 \pm 0.77 \cdot 10^{-4}$**	**$1.94 \pm 1.14 \cdot 10^{-3}$**	$7.39 \pm 5.33 \cdot 10^{-5}$	$1.10 \pm 0.79 \cdot 10^{-3}$
	TabSyn	$9.26 \pm 2.41 \cdot 10^{-4}$	$1.14 \pm 0.36 \cdot 10^{-2}$	$1.22 \pm 0.65 \cdot 10^{-5}$	$1.81 \pm 0.96 \cdot 10^{-4}$
	TVAE	$3.37 \pm 0.24 \cdot 10^{-3}$	$5.00 \pm 0.36 \cdot 10^{-2}$	**$1.78 \pm 0.43 \cdot 10^{-7}$**	**$2.65 \pm 0.64 \cdot 10^{-6}$**
	SMOTE	-	-	$2.44 \pm 0.52 \cdot 10^{-7}$	$3.63 \pm 0.78 \cdot 10^{-6}$
	UC-SMOTE	-	-	$1.03 \pm 0.12 \cdot 10^{-6}$	$1.53 \pm 0.18 \cdot 10^{-5}$

Table D.13: CO_2 emissions (in Kg) and Energy Consumption (in kWh) of the benchmark challengers. The energy consumption is obtained by summing the CPU, GPU and RAM energy. Training costs are given for all models on the same basis of 400 epochs. Sampling costs are given for 5 samples. The best values are formatted in **bold green** and the worse are in red.

Appendix D.2. Whole Tuning Cost Estimation

As mentioned in Section 4.2 we could not perform the hyperparameter search and training phases on a uniform hardware and software architecture and we estimated the tuning cost with Equation (1).

Each trial can be stopped based on three conditions: an early stopping decided by the model, a poor C2ST performance or a time limit. Therefore, to get an accurate estimate of the init-cost and the cost-per-step from the single-GPU mentioned in Appendix D.1, we needed to extract from our logs the effective number of training steps performed per trial.

In addition, we also considered the parallelization scheme applied during the tuning procedure. One issue with TabSyn was that we observed a general slowdown when the model was parallelized too heavily. This issue was even more marked on datasets with a large number of columns. We hence reduced the number trials per GPU and the global number of trials to 100 to fall back to a reasonable time for this model.

Finally, we measured cost on a typical configuration used during our tuning experiment: 8 Tesla V100-SXM2 32 GB. Considering those 8 GPUs, we used the following parallel allocation of trials: 64 for TVAE and CTGAN, 40 for TabDDPM, and 16 for TabSyn. A model that can be easily parallelized during the hyperparameter tuning phase offers a cost advantage. It is hence

important to consider this aspect during the evaluation process. The results are presented in Table D.14.

Dataset	Model	Metrics		
		Energy (kWh) ↓	Emissions (Kg) ↓	Duration (HH:MM) ↓
Abalone	TVAE	0.00	0.01	00:02
	CTGAN	0.01	0.08	00:19
	TabDDPM	**0.00**	0.05	00:10
	TabSyn	0.03	0.40	01:31
	SMOTE	0.00	**0.00**	00:00
	UC-SMOTE	0.00	0.00	**00:00**
Adult	TVAE	0.01	0.17	00:42
	CTGAN	0.02	0.30	01:15
	TabDDPM	0.01	0.12	00:25
	TabSyn	0.08	0.91	03:27
	SMOTE	**0.00**	**0.01**	**00:04**
	UC-SMOTE	0.00	0.02	00:06
Bank marketing	TVAE	0.01	0.10	00:25
	CTGAN	0.03	0.38	01:35
	TabDDPM	0.01	0.13	00:29
	TabSyn	0.08	0.96	03:27
	SMOTE	**0.00**	**0.02**	00:06
	UC-SMOTE	0.00	0.02	**00:05**
Bike sharing	TVAE	**0.00**	0.05	00:12
	CTGAN	0.03	0.40	01:41
	TabDDPM	0.01	0.12	00:22
	TabSyn	0.05	0.61	02:18
	SMOTE	0.00	**0.00**	00:00
	UC-SMOTE	0.00	0.00	**00:00**
Black friday	TVAE	0.03	0.46	01:55
	CTGAN	0.08	1.18	04:54
	TabDDPM	0.01	0.13	00:23
	TabSyn	0.21	2.40	09:19
	SMOTE	**0.00**	**0.06**	**00:19**
	UC-SMOTE	0.01	0.22	01:07
Cardio	TVAE	0.01	0.16	00:41
	CTGAN	0.03	0.51	02:07
	TabDDPM	0.01	0.08	00:12
	TabSyn	0.10	1.38	04:13
	SMOTE	**0.00**	**0.00**	**00:00**
	UC-SMOTE	0.00	0.00	00:00
Churn	TVAE	**0.00**	0.03	00:07
	CTGAN	0.02	0.29	01:11
	TabDDPM	0.02	0.31	00:48
	TabSyn	0.04	0.53	02:02
	SMOTE	0.00	**0.00**	00:00
	UC-SMOTE	0.00	0.00	**00:00**

Tuning costs estimation (continued).

Dataset	Model	Metrics		
		Energy (kWh) ↓	Emissions (Kg) ↓	Duration (HH:MM) ↓
Covertype	TVAE	0.19	2.77	11:34
	CTGAN	0.54	8.04	33:17
	TabDDPM	0.03	0.49	01:48
	TabSyn	0.19	1.53	07:54
	SMOTE	**0.02**	**0.26**	**01:19**
	UC-SMOTE	0.02	0.28	01:27
Diamonds	TVAE	0.01	0.10	00:25
	CTGAN	0.03	0.40	01:39
	TabDDPM	0.01	0.15	00:24
	TabSyn	0.08	1.01	03:22
	SMOTE	**0.00**	0.01	00:02
	UC-SMOTE	0.00	**0.00**	**00:00**
Heloc	TVAE	0.00	0.04	00:10
	CTGAN	0.03	0.49	02:02
	TabDDPM	**0.00**	0.03	00:05
	TabSyn	0.04	0.54	02:08
	SMOTE	0.00	**0.00**	**00:00**
	UC-SMOTE	0.00	0.00	00:00
Higgs	TVAE	0.04	0.53	02:13
	CTGAN	0.11	1.61	06:42
	TabDDPM	0.01	0.10	00:15
	TabSyn	0.18	1.58	07:50
	SMOTE	**0.00**	**0.00**	**00:00**
	UC-SMOTE	0.00	0.00	00:01
House 16h	TVAE	0.01	0.09	00:23
	CTGAN	0.01	0.10	00:25
	TabDDPM	**0.00**	0.04	00:06
	TabSyn	0.05	0.58	02:16
	SMOTE	0.00	**0.00**	**00:00**
	UC-SMOTE	0.00	0.00	00:00
Insurance	TVAE	0.00	**0.00**	00:00
	CTGAN	**0.00**	0.03	00:06
	TabDDPM	0.00	0.05	00:12
	TabSyn	0.03	0.35	01:25
	SMOTE	0.00	0.00	00:00
	UC-SMOTE	0.00	0.00	**00:00**
King	TVAE	0.01	0.09	00:23
	CTGAN	0.01	0.21	00:51
	TabDDPM	0.01	0.17	00:27
	TabSyn	0.05	0.66	02:26
	SMOTE	**0.00**	**0.00**	00:01
	UC-SMOTE	0.00	0.00	**00:00**

Tuning costs estimation (continued).

Dataset	Model	Metrics		
		Energy (kWh) ↓	Emissions (Kg) ↓	Duration (HH:MM) ↓
Miniboone	TVAE	0.04	0.56	02:19
	CTGAN	0.05	0.73	02:59
	TabDDPM	0.01	0.10	00:14
	TabSyn	0.31	2.02	11:44
	SMOTE	**0.00**	**0.00**	**00:00**
	UC-SMOTE	0.00	0.01	00:02
Moons	TVAE	0.01	0.08	00:20
	CTGAN	0.02	0.29	01:12
	TabDDPM	0.01	0.08	00:12
	TabSyn	0.05	0.87	02:35
	SMOTE	**0.00**	**0.00**	**00:00**
	UC-SMOTE	0.00	0.00	00:00

Table D.14: Estimated hyperparameter search cost based on the estimated training cost of the best models presented in Table D.13. All costs associated with TabSyn include those incurred by the VAE and the denoiser. The energy and emissions values are rounded to two decimals and take into account the number of trials we could run in parallel per model. The best values per metric are formatted in **bold green** and the worse are in red.

Appendix E. Base Models and their Tuned Versions

We also trained the models using their native codes and hyperparameters to provide an additional reference for comparison with their tuned versions. Table E.15 presents the results as evaluated under the same procedure as the tuned models for C2ST, DCR-Rate, ML-Efficacy, *column-wise similarity* (named "Shape"), and *pair-wise correlation* (named "Pair"). For TabDDPM, since there is no default hyperparameters provided [15], we fixed one based on the base configuration provided in the authors' GitHub repository.

Dataset	Model	Metrics				
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑
Abalone	Train Copy	0.51 ± 0.00	1.00 ± 0.00	0.23 ± 0.01	0.96 ± 0.01	0.88 ± 0.01
	CTGAN	0.99 ± 0.01	**0.60 ± 0.01**	0.12 ± 0.03	0.87 ± 0.02	0.76 ± 0.01
	TVAE	0.96 ± 0.00	0.61 ± 0.02	0.22 ± 0.02	0.91 ± 0.02	0.83 ± 0.03
	TabDDPM	1.00 ± 0.00	0.64 ± 0.01	0.00 ± 0.00	0.85 ± 0.01	0.70 ± 0.01
	TabSyn	**0.78 ± 0.02**	0.64 ± 0.01	**0.22 ± 0.01**	**0.95 ± 0.01**	**0.88 ± 0.01**
Adult	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.71 ± 0.01	0.99 ± 0.00	0.98 ± 0.00
	CTGAN	0.96 ± 0.01	0.61 ± 0.01	0.61 ± 0.05	0.88 ± 0.02	0.82 ± 0.02
	TVAE	0.94 ± 0.01	**0.61 ± 0.00**	0.63 ± 0.03	0.92 ± 0.00	0.85 ± 0.01
	TabDDPM	**0.66 ± 0.01**	0.62 ± 0.00	**0.67 ± 0.00**	**0.98 ± 0.00**	**0.95 ± 0.00**

Results per datasets and models under diverse metrics for the *base models* (continued).

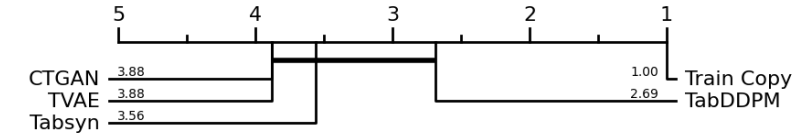
Dataset	Model	Metrics				
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑
	TabSyn	0.71 ± 0.06	0.62 ± 0.00	0.66 ± 0.01	0.98 ± 0.01	0.95 ± 0.02
Bank marketing	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.54 ± 0.01	0.99 ± 0.00	0.98 ± 0.01
	CTGAN	0.89 ± 0.01	**0.62 ± 0.01**	0.33 ± 0.05	0.92 ± 0.00	0.86 ± 0.01
	TVAE	0.95 ± 0.00	0.62 ± 0.00	**0.52 ± 0.02**	0.91 ± 0.01	0.84 ± 0.02
	TabDDPM	0.68 ± 0.01	0.63 ± 0.01	0.48 ± 0.02	**0.99 ± 0.01**	**0.96 ± 0.00**
	TabSyn	**0.65 ± 0.04**	0.63 ± 0.01	0.47 ± 0.01	0.98 ± 0.01	0.96 ± 0.00
Bike sharing	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.94 ± 0.00	0.99 ± 0.00	0.53 ± 0.00
	CTGAN	1.00 ± 0.00	**0.61 ± 0.00**	0.37 ± 0.03	0.93 ± 0.01	0.51 ± 0.01
	TVAE	1.00 ± 0.00	0.61 ± 0.01	0.42 ± 0.08	0.91 ± 0.01	0.51 ± 0.01
	TabDDPM	**0.85 ± 0.01**	0.61 ± 0.00	**0.67 ± 0.03**	**0.98 ± 0.00**	**0.53 ± 0.00**
	TabSyn	0.94 ± 0.03	0.61 ± 0.00	0.31 ± 0.04	0.95 ± 0.01	0.52 ± 0.01
Black friday	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.53 ± 0.01	1.00 ± 0.00	0.99 ± 0.00
	CTGAN	0.92 ± 0.01	**0.66 ± 0.01**	**0.40 ± 0.01**	0.94 ± 0.01	0.84 ± 0.01
	TVAE	0.98 ± 0.00	0.67 ± 0.01	0.29 ± 0.08	0.83 ± 0.01	0.73 ± 0.01
	TabDDPM	0.92 ± 0.00	0.67 ± 0.01	0.31 ± 0.03	**0.99 ± 0.00**	**0.98 ± 0.00**
	TabSyn	**0.91 ± 0.01**	0.68 ± 0.00	0.13 ± 0.03	0.98 ± 0.00	0.96 ± 0.01
Cardio	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.72 ± 0.01	1.00 ± 0.00	0.98 ± 0.01
	CTGAN	1.00 ± 0.01	**0.62 ± 0.01**	0.67 ± 0.02	0.93 ± 0.01	0.93 ± 0.01
	TVAE	1.00 ± 0.00	0.63 ± 0.00	0.67 ± 0.04	0.88 ± 0.01	0.90 ± 0.02
	TabDDPM	0.59 ± 0.00	0.64 ± 0.00	**0.72 ± 0.01**	**0.99 ± 0.00**	0.96 ± 0.02
	TabSyn	**0.58 ± 0.01**	0.64 ± 0.01	0.72 ± 0.01	0.99 ± 0.01	**0.97 ± 0.01**
Churn	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.59 ± 0.02	0.95 ± 0.00	0.87 ± 0.01
	CTGAN	0.87 ± 0.02	**0.60 ± 0.00**	0.19 ± 0.10	0.85 ± 0.02	0.79 ± 0.00
	TVAE	0.98 ± 0.01	0.60 ± 0.01	**0.45 ± 0.10**	0.75 ± 0.03	0.63 ± 0.03
	TabDDPM	1.00 ± 0.00	0.65 ± 0.40	0.00 ± 0.00	0.59 ± 0.02	0.56 ± 0.02
	TabSyn	**0.64 ± 0.03**	0.61 ± 0.01	0.43 ± 0.13	**0.92 ± 0.01**	**0.85 ± 0.00**
Covertype	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.90 ± 0.00	1.00 ± 0.00	1.00 ± 0.01
	CTGAN	1.00 ± 0.00	**0.60 ± 0.00**	0.66 ± 0.01	0.97 ± 0.00	0.94 ± 0.01
	TVAE	1.00 ± 0.00	0.60 ± 0.00	0.72 ± 0.01	0.98 ± 0.00	0.95 ± 0.01
	TabDDPM	0.82 ± 0.01	0.60 ± 0.00	0.73 ± 0.01	**1.00 ± 0.00**	**0.99 ± 0.00**
	TabSyn	**0.72 ± 0.04**	0.60 ± 0.00	**0.79 ± 0.03**	0.99 ± 0.00	0.99 ± 0.00
Diamonds	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.98 ± 0.00	0.99 ± 0.00	0.77 ± 0.01
	CTGAN	0.98 ± 0.00	**0.60 ± 0.00**	0.87 ± 0.01	0.91 ± 0.01	0.75 ± 0.04
	TVAE	0.97 ± 0.01	0.60 ± 0.01	0.91 ± 0.01	0.92 ± 0.02	**0.76 ± 0.03**
	TabDDPM	**0.76 ± 0.01**	0.61 ± 0.00	**0.97 ± 0.00**	**0.99 ± 0.01**	0.71 ± 0.01
	TabSyn	0.95 ± 0.02	0.64 ± 0.00	0.19 ± 0.06	0.96 ± 0.00	0.72 ± 0.01
Heloc	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.70 ± 0.01	0.99 ± 0.00	0.97 ± 0.01
	CTGAN	1.00 ± 0.00	**0.63 ± 0.00**	0.40 ± 0.14	0.87 ± 0.05	0.87 ± 0.07
	TVAE	0.98 ± 0.01	0.63 ± 0.02	0.69 ± 0.01	0.90 ± 0.00	0.75 ± 0.01
	TabDDPM	**0.71 ± 0.01**	0.64 ± 0.02	**0.70 ± 0.01**	**0.97 ± 0.00**	0.94 ± 0.01
	TabSyn	0.75 ± 0.01	0.65 ± 0.01	0.69 ± 0.01	0.97 ± 0.00	**0.96 ± 0.01**
Higgs	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.74 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	CTGAN	1.00 ± 0.00	**0.60 ± 0.00**	0.61 ± 0.08	0.87 ± 0.02	0.95 ± 0.00
	TVAE	1.00 ± 0.00	0.61 ± 0.02	0.68 ± 0.02	0.78 ± 0.00	0.94 ± 0.01
	TabDDPM	0.79 ± 0.02	0.60 ± 0.01	0.72 ± 0.01	**0.98 ± 0.01**	0.93 ± 0.02
	TabSyn	**0.59 ± 0.01**	0.61 ± 0.00	**0.73 ± 0.00**	0.94 ± 0.00	**0.99 ± 0.00**

Results per datasets and models under diverse metrics for the *base models* (continued).

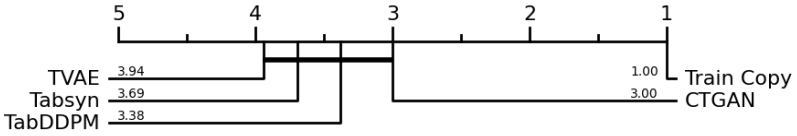
Dataset	Model	Metrics				
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑
House 16h	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.64 ± 0.01	0.99 ± 0.01	0.99 ± 0.00
	CTGAN	1.00 ± 0.00	**0.61 ± 0.02**	0.20 ± 0.05	0.86 ± 0.01	0.95 ± 0.01
	TVAE	0.98 ± 0.00	0.61 ± 0.01	0.36 ± 0.02	0.89 ± 0.00	0.96 ± 0.01
	TabDDPM	**0.63 ± 0.01**	0.61 ± 0.01	**0.57 ± 0.01**	**0.98 ± 0.00**	0.97 ± 0.01
	TabSyn	0.84 ± 0.03	0.62 ± 0.00	0.33 ± 0.06	0.96 ± 0.01	**0.98 ± 0.01**
Insurance	Train Copy	0.48 ± 0.01	1.00 ± 0.00	0.85 ± 0.03	0.96 ± 0.01	0.91 ± 0.01
	CTGAN	0.92 ± 0.01	0.67 ± 0.06	−0.18 ± 0.05	0.84 ± 0.00	0.81 ± 0.01
	TVAE	0.87 ± 0.02	0.62 ± 0.02	0.61 ± 0.03	0.85 ± 0.03	0.77 ± 0.03
	TabDDPM	**0.58 ± 0.01**	**0.60 ± 0.02**	**0.83 ± 0.02**	**0.95 ± 0.01**	**0.90 ± 0.01**
	TabSyn	0.60 ± 0.03	0.61 ± 0.02	0.83 ± 0.01	0.94 ± 0.01	0.89 ± 0.01
King	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.87 ± 0.00	0.98 ± 0.01	0.97 ± 0.01
	CTGAN	1.00 ± 0.00	**0.60 ± 0.01**	0.54 ± 0.04	0.89 ± 0.01	0.93 ± 0.00
	TVAE	0.99 ± 0.01	0.60 ± 0.00	**0.70 ± 0.02**	0.92 ± 0.01	0.91 ± 0.01
	TabDDPM	1.00 ± 0.00	0.91 ± 0.14	−192.96 ± 153.37	0.33 ± 0.03	0.74 ± 0.02
	TabSyn	**0.97 ± 0.01**	0.62 ± 0.01	0.05 ± 0.04	**0.96 ± 0.01**	**0.94 ± 0.00**
Miniboone	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.89 ± 0.01	0.99 ± 0.00	0.99 ± 0.01
	CTGAN	1.00 ± 0.00	**0.60 ± 0.00**	0.54 ± 0.05	0.80 ± 0.03	0.56 ± 0.01
	TVAE	1.00 ± 0.00	0.60 ± 0.00	0.82 ± 0.01	0.91 ± 0.02	0.56 ± 0.00
	TabDDPM	0.82 ± 0.01	0.63 ± 0.01	0.88 ± 0.00	0.95 ± 0.02	0.83 ± 0.03
	TabSyn	**0.71 ± 0.09**	0.60 ± 0.00	**0.89 ± 0.01**	**0.98 ± 0.01**	**0.89 ± 0.05**
Moons	Train Copy	0.50 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00
	CTGAN	0.93 ± 0.01	0.60 ± 0.05	**1.00 ± 0.00**	0.94 ± 0.01	0.70 ± 0.03
	TVAE	0.82 ± 0.02	0.61 ± 0.03	1.00 ± 0.00	0.97 ± 0.01	0.76 ± 0.02
	TabDDPM	0.61 ± 0.16	**0.53 ± 0.15**	0.97 ± 0.06	0.93 ± 0.10	0.90 ± 0.16
	TabSyn	**0.58 ± 0.03**	0.61 ± 0.01	1.00 ± 0.00	**0.98 ± 0.01**	**0.98 ± 0.01**

Table E.15: Results for ***base models***. Models are trained using their default hyperparameters as provided by the authors in their papers. The best values per metric are formatted in **bold green** and the worse values are in red.

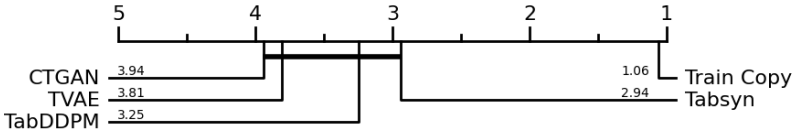
Appendix F. Quick Search on Reduced Hyperparameter Space



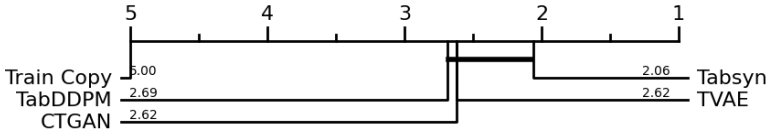
(a) C2ST



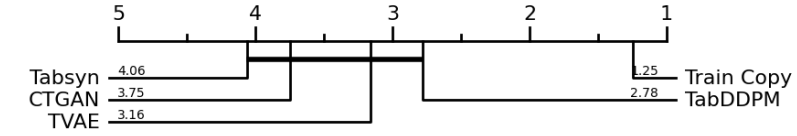
(b) Column-wise similarity



(c) Pair-wise correlation



(d) DCR-Rate



(e) ML-Efficacy

Figure F.12: Models' ranking under the *light* (limited-budget) experiment setup with critical difference diagrams on C2ST, *column-wise similarity*, *pair-wise correlation*, DCR-Rate, and ML-Efficacy metrics over all datasets.

With the reduced hyperparameters search spaces presented in Appendix A, we ran a limited-budget hyperparameter tuning described in Section 6. The experiment was done on all dataset (Table 2). We performed 50 trials per

fold with 3 folds, meaning we ran a total of 150 trials for this limited-budget experiment. The results par datasets are shown in Table F.16.

Dataset	Model	Metrics					
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Sampling time ↓
Abalone	Train Copy	0.51 ± 0.00	1.00 ± 0.00	0.23 ± 0.01	0.96 ± 0.01	0.88 ± 0.01	-
	CTGAN	0.71 ± 0.03	0.63 ± 0.00	0.17 ± 0.01	0.93 ± 0.02	0.86 ± 0.04	00 ± 0.01
	TVAE	**0.64 ± 0.02**	0.67 ± 0.00	0.23 ± 0.01	0.93 ± 0.02	**0.89 ± 0.03**	**00 ± 0.00**
	TabDDPM	0.78 ± 0.01	0.69 ± 0.03	**0.23 ± 0.01**	**0.95 ± 0.00**	0.88 ± 0.02	03 ± 0.88
	TabSyn	0.80 ± 0.00	**0.62 ± 0.01**	0.19 ± 0.03	0.93 ± 0.01	0.85 ± 0.01	00 ± 0.01
Adult	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.71 ± 0.01	0.99 ± 0.00	0.98 ± 0.00	-
	CTGAN	0.77 ± 0.01	0.72 ± 0.00	0.65 ± 0.00	0.96 ± 0.01	0.89 ± 0.02	00 ± 0.02
	TVAE	0.77 ± 0.01	0.72 ± 0.01	0.65 ± 0.02	0.96 ± 0.01	0.93 ± 0.00	**00 ± 0.00**
	TabDDPM	**0.67 ± 0.01**	**0.62 ± 0.00**	**0.67 ± 0.01**	**0.97 ± 0.01**	**0.94 ± 0.01**	11 ± 0.63
	TabSyn	0.73 ± 0.01	0.62 ± 0.00	0.66 ± 0.01	0.97 ± 0.01	0.94 ± 0.01	02 ± 0.01
Bank marketing	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.54 ± 0.01	0.99 ± 0.00	0.98 ± 0.01	-
	CTGAN	0.72 ± 0.01	0.63 ± 0.00	0.45 ± 0.04	**0.98 ± 0.00**	0.94 ± 0.01	00 ± 0.01
	TVAE	0.78 ± 0.02	0.63 ± 0.00	0.47 ± 0.03	0.97 ± 0.01	0.94 ± 0.00	**00 ± 0.00**
	TabDDPM	**0.68 ± 0.00**	0.63 ± 0.01	**0.52 ± 0.03**	0.97 ± 0.00	**0.95 ± 0.01**	16 ± 2.18
	TabSyn	0.75 ± 0.00	**0.62 ± 0.00**	0.42 ± 0.04	0.97 ± 0.01	0.95 ± 0.00	02 ± 0.01
Bike sharing	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.94 ± 0.00	0.99 ± 0.00	0.53 ± 0.00	-
	CTGAN	0.89 ± 0.01	0.61 ± 0.01	0.71 ± 0.01	**0.98 ± 0.01**	**0.53 ± 0.00**	00 ± 0.00
	TVAE	0.81 ± 0.01	0.61 ± 0.01	0.69 ± 0.03	0.96 ± 0.00	0.53 ± 0.00	**00 ± 0.00**
	TabDDPM	**0.80 ± 0.02**	0.63 ± 0.01	**0.83 ± 0.01**	0.96 ± 0.02	0.53 ± 0.01	06 ± 1.34
	TabSyn	0.91 ± 0.01	**0.61 ± 0.01**	0.32 ± 0.06	0.95 ± 0.01	0.52 ± 0.00	00 ± 0.01
Black friday	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.53 ± 0.01	1.00 ± 0.00	0.99 ± 0.00	-
	CTGAN	**0.87 ± 0.01**	0.68 ± 0.00	0.36 ± 0.01	**0.97 ± 0.01**	0.91 ± 0.02	01 ± 0.17
	TVAE	0.95 ± 0.01	0.68 ± 0.00	0.27 ± 0.02	0.97 ± 0.00	0.95 ± 0.00	**00 ± 0.02**
	TabDDPM	0.89 ± 0.01	**0.67 ± 0.00**	**0.44 ± 0.02**	0.97 ± 0.01	**0.96 ± 0.01**	50 ± 17.19
	TabSyn	0.89 ± 0.01	0.68 ± 0.00	0.16 ± 0.01	0.97 ± 0.01	0.95 ± 0.01	07 ± 0.01
Cardio	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.72 ± 0.01	1.00 ± 0.00	0.98 ± 0.01	-
	CTGAN	0.64 ± 0.01	0.64 ± 0.01	0.72 ± 0.01	**0.99 ± 0.00**	0.96 ± 0.00	00 ± 0.06
	TVAE	0.71 ± 0.01	0.64 ± 0.01	**0.73 ± 0.01**	0.96 ± 0.01	0.96 ± 0.01	**00 ± 0.01**
	TabDDPM	**0.57 ± 0.01**	**0.64 ± 0.00**	0.72 ± 0.01	0.99 ± 0.00	**0.98 ± 0.01**	12 ± 4.58
	TabSyn	0.59 ± 0.01	0.64 ± 0.00	0.72 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	03 ± 0.01
Churn	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.59 ± 0.02	0.95 ± 0.00	0.87 ± 0.01	-
	CTGAN	**0.63 ± 0.01**	0.63 ± 0.01	0.39 ± 0.01	**0.93 ± 0.00**	**0.84 ± 0.01**	00 ± 0.01
	TVAE	0.64 ± 0.00	0.63 ± 0.01	**0.53 ± 0.01**	0.92 ± 0.00	0.84 ± 0.00	**00 ± 0.00**
	TabDDPM	0.98 ± 0.03	0.67 ± 0.18	0.02 ± 0.03	0.59 ± 0.05	0.55 ± 0.02	18 ± 1.06
	TabSyn	0.68 ± 0.06	**0.61 ± 0.00**	0.46 ± 0.03	0.91 ± 0.01	0.84 ± 0.01	00 ± 0.01

Results per datasets and models under diverse metrics for the limited-budget search (continued).

Dataset	Model	Metrics					
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Sampling time ↓
Covertype	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.90 ± 0.00	1.00 ± 0.00	1.00 ± 0.01	-
	CTGAN	0.98 ± 0.00	**0.60 ± 0.00**	0.69 ± 0.00	**0.98 ± 0.00**	0.95 ± 0.01	06 ± 1.58
	TVAE	0.90 ± 0.01	0.60 ± 0.00	**0.77 ± 0.01**	0.98 ± 0.00	**0.96 ± 0.00**	02 ± 0.07
	TabDDPM	0.97 ± 0.01	0.64 ± 0.02	0.69 ± 0.00	0.94 ± 0.02	0.89 ± 0.03	355 ± 10.90
	TabSyn	**0.87 ± 0.00**	0.61 ± 0.01	0.65 ± 0.03	0.98 ± 0.00	0.68 ± 0.00	**02 ± 0.02**
Diamonds	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.98 ± 0.00	0.99 ± 0.00	0.77 ± 0.01	-
	CTGAN	0.89 ± 0.01	0.64 ± 0.01	0.94 ± 0.00	0.95 ± 0.00	**0.74 ± 0.02**	00 ± 0.08
	TVAE	0.76 ± 0.02	0.64 ± 0.00	0.96 ± 0.01	0.95 ± 0.01	0.73 ± 0.01	**00 ± 0.01**
	TabDDPM	**0.75 ± 0.01**	**0.61 ± 0.01**	**0.97 ± 0.00**	**0.97 ± 0.01**	0.70 ± 0.02	10 ± 1.26
	TabSyn	0.92 ± 0.01	0.64 ± 0.01	0.65 ± 0.09	0.95 ± 0.01	0.72 ± 0.02	02 ± 0.01
Heloc	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.70 ± 0.01	0.99 ± 0.00	0.97 ± 0.01	-
	CTGAN	0.93 ± 0.02	0.64 ± 0.01	**0.70 ± 0.01**	**0.97 ± 0.00**	0.93 ± 0.01	00 ± 0.02
	TVAE	0.92 ± 0.01	**0.63 ± 0.01**	0.69 ± 0.02	0.94 ± 0.00	0.88 ± 0.01	**00 ± 0.00**
	TabDDPM	**0.71 ± 0.02**	0.67 ± 0.01	0.69 ± 0.02	0.95 ± 0.01	0.92 ± 0.03	03 ± 1.99
	TabSyn	0.82 ± 0.01	0.65 ± 0.01	0.69 ± 0.01	0.95 ± 0.01	**0.95 ± 0.01**	00 ± 0.01
Higgs	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.74 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	-
	CTGAN	0.87 ± 0.00	**0.60 ± 0.01**	0.70 ± 0.01	**0.98 ± 0.00**	**0.98 ± 0.00**	00 ± 0.09
	TVAE	0.92 ± 0.01	0.60 ± 0.01	0.71 ± 0.02	0.94 ± 0.01	0.97 ± 0.00	**00 ± 0.03**
	TabDDPM	**0.61 ± 0.02**	0.62 ± 0.01	**0.73 ± 0.00**	0.98 ± 0.01	0.95 ± 0.01	08 ± 1.00
	TabSyn	0.76 ± 0.05	0.60 ± 0.01	0.71 ± 0.02	0.92 ± 0.01	0.98 ± 0.00	04 ± 0.01
House 16h	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.64 ± 0.01	0.99 ± 0.01	0.99 ± 0.00	-
	CTGAN	0.84 ± 0.02	0.62 ± 0.00	0.45 ± 0.02	**0.97 ± 0.01**	0.97 ± 0.01	**00 ± 0.01**
	TVAE	0.84 ± 0.00	0.62 ± 0.01	0.46 ± 0.03	0.94 ± 0.00	0.98 ± 0.00	00 ± 0.00
	TabDDPM	**0.64 ± 0.01**	**0.61 ± 0.01**	**0.61 ± 0.01**	0.96 ± 0.01	0.94 ± 0.03	02 ± 0.75
	TabSyn	0.83 ± 0.03	0.62 ± 0.01	0.35 ± 0.01	0.95 ± 0.01	**0.99 ± 0.01**	01 ± 0.01
Insurance	Train Copy	0.48 ± 0.01	1.00 ± 0.00	0.85 ± 0.03	0.96 ± 0.01	0.91 ± 0.01	-
	CTGAN	0.67 ± 0.02	0.92 ± 0.01	0.66 ± 0.04	**0.94 ± 0.01**	0.85 ± 0.02	00 ± 0.01
	TVAE	0.65 ± 0.02	0.91 ± 0.02	0.80 ± 0.03	0.93 ± 0.01	0.88 ± 0.01	**00 ± 0.00**
	TabDDPM	**0.61 ± 0.02**	0.62 ± 0.02	**0.83 ± 0.03**	0.94 ± 0.01	**0.89 ± 0.02**	06 ± 2.76
	TabSyn	0.63 ± 0.02	**0.60 ± 0.03**	0.80 ± 0.01	0.93 ± 0.01	0.88 ± 0.02	00 ± 0.01
King	Train Copy	0.50 ± 0.01	1.00 ± 0.00	0.87 ± 0.00	0.98 ± 0.01	0.97 ± 0.01	-
	CTGAN	0.96 ± 0.01	0.62 ± 0.00	0.66 ± 0.03	**0.97 ± 0.00**	**0.94 ± 0.01**	00 ± 0.13
	TVAE	**0.95 ± 0.01**	**0.61 ± 0.01**	**0.80 ± 0.01**	0.95 ± 0.01	0.94 ± 0.00	**00 ± 0.01**
	TabDDPM	1.00 ± 0.00	0.70 ± 0.46	0.03 ± 0.68	0.28 ± 0.07	0.76 ± 0.00	05 ± 0.43
	TabSyn	0.98 ± 0.01	0.62 ± 0.01	−0.01 ± 0.01	0.94 ± 0.01	0.93 ± 0.00	01 ± 0.01
Miniboone	Train Copy	0.50 ± 0.00	1.00 ± 0.00	0.89 ± 0.01	0.99 ± 0.00	0.99 ± 0.01	-
	CTGAN	0.90 ± 0.01	**0.60 ± 0.01**	0.87 ± 0.01	0.93 ± 0.01	0.58 ± 0.01	01 ± 0.03
	TVAE	0.94 ± 0.01	0.60 ± 0.00	0.87 ± 0.00	0.94 ± 0.00	0.59 ± 0.01	**00 ± 0.03**
	TabDDPM	**0.73 ± 0.01**	0.62 ± 0.00	**0.89 ± 0.01**	0.96 ± 0.01	**0.83 ± 0.05**	12 ± 1.35
	TabSyn	0.87 ± 0.01	0.60 ± 0.01	0.87 ± 0.00	**0.96 ± 0.00**	0.76 ± 0.14	06 ± 0.08

Results per datasets and models under diverse metrics for the limited-budget search (continued).

Dataset	Model	Metrics					
		C2ST ↓	DCR-Rate ↓	ML-Efficacy ↑	Shape ↑	Pair ↑	Sampling time ↓
Moons	Train Copy	0.50 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	-
	CTGAN	0.70 ± 0.02	0.62 ± 0.02	**1.00 ± 0.00**	0.96 ± 0.01	0.96 ± 0.01	00 ± 0.02
	TVAE	0.69 ± 0.01	0.62 ± 0.01	1.00 ± 0.00	0.96 ± 0.01	0.93 ± 0.02	**00 ± 0.00**
	TabDDPM	**0.53 ± 0.01**	**0.61 ± 0.01**	1.00 ± 0.00	**0.99 ± 0.00**	**0.98 ± 0.01**	08 ± 2.13
	TabSyn	0.59 ± 0.01	0.61 ± 0.01	1.00 ± 0.00	0.98 ± 0.01	0.97 ± 0.01	01 ± 0.00

Table F.16: Limited-budget experiment results under various metrics. Results are averaged over 3 folds with 5 synthetic samples per fold as done in the extensive hyperparameter tuning. The best values per metric are formatted in **bold green** and the worse values are in red.