A Sobering Look at Tabular Data Generation via Probabilistic Circuits

Davide Scassola[2, 3, *] Dylan Ponsford[1, *] Adrián Javaloy[1] Sebastiano Saccani[3] Luca Bortolussi[2]
Henry Gouk[1] Antonio Vergari[1]

[1]School of Informatics, University of Edinburgh, Edinburgh, UK
[2]AILAB, University of Trieste, Trieste, Italy
[3]Aindo SpA, AREA Science Park, Trieste, Italy,

Abstract

Tabular data is more challenging to generate than text and images, due to its heterogeneous features and much lower sample sizes. On this task, diffusion-based models are the current state-of-the-art (SotA) model class, achieving almost perfect performance on commonly used benchmarks. In this paper, *we question the perception of progress for tabular data generation*. First, we highlight the limitations of current protocols to evaluate the fidelity of generated data, and advocate for alternative ones. Next, we revisit a simple baseline—hierarchical mixture models in the form of deep probabilistic circuits (PCs)—which delivers competitive or superior performance to SotA models *for a fraction of the cost*. PCs are the generative counterpart of decision forests, and as such can natively handle heterogeneous data as well as deliver tractable probabilistic generation and inference. Finally, in a rigorous empirical analysis we show that the apparent saturation of progress for SotA models is largely due to the use of inadequate metrics. As such, we highlight that there is still much to be done to generate realistic tabular data. Code available at https://github.com/april-tools/tabpc.

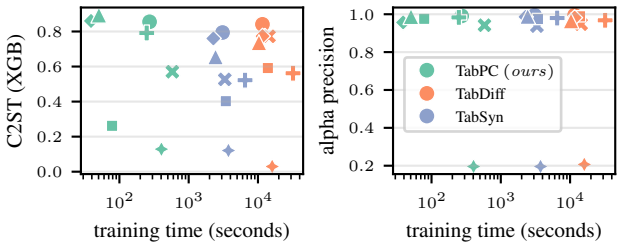


Figure 1: **PCs for tabular data (TABPC) compete with diffusion-based approaches at a fraction of the cost** for all datasets (denoted by marker shape, see Fig. 2) on fidelity metrics such as α-PRECISION [Alaa et al., 2022] and C2ST [Lopez-Paz and Oquab, 2017] when computed with an XGBoost classifier. We remark this is not the commonly used implementation of C2ST, which instead uses a logistic regressor for which even a fully-factorised model yields SotA results, **delivering a false sense of progress** (see Fig. 3).

1 INTRODUCTION

Deep Generative Models (DGMs; Tomczak 2024) have mastered generating high-dimensional and structured data such as images [Croitoru et al., 2023] and text [Achiam et al., 2023]. However, a modality still considered challenging for DGMs is *tabular data*—data found in the rows (samples) and columns (features) of a database table [Borisov et al., 2022]. One explanation for this is that tabular data can be *heterogeneous*, i.e., be both continuous and discrete, and have different statistical data types [Valera and Ghahramani, 2017]. Moreover, tables usually have *fewer samples* compared to datasets typically used for training neural networks. These issues also explain why simple *discriminative* models such as decision trees and their ensembles [Chen and Guestrin, 2016] can still outperform neural predictors on tabular data [Grinsztajn et al., 2022, Shwartz-Ziv and Armon, 2022, Van Breugel and Van Der Schaar, 2024].

Despite this, recent diffusion-based DGMs [Kotelnikov et al., 2023, Zhang et al., 2024, Shi et al., 2025, Guzmán-Cordero et al., 2025] have shown remarkable progress on a number of commonly used tabular data generation (TDG) benchmarks [Becker and Kohavi, 1996, Bock, 2004, Yeh, 2009, Chen, 2015], to the point that metrics used to measure the *fidelity* of generated data [Alaa et al., 2022, Dat, 2025, Kim et al., 2023] cannot be further improved. For example, the *classifier two-sample test* (C2ST; Lopez-Paz and Oquab 2017) metric, which uses a classifier to distinguish generated from real data, or the TREND metric [Dat, 2025] , used

[*]Equal contribution (author order was randomized).

to measure dependency between two features, can be close to 1.0 (their maximum value, i.e., 100% fidelity) for SotA diffusion models [Shi et al., 2025].

In this paper, *we question the apparent progress of DGMs for TDG* via a twofold approach. First, we investigate some of the most commonly used fidelity metrics and their implementation. We note that the saturated performance of DGMs can be explained by simplistic choices of how metrics are computed, e.g., using linear classifiers for C2ST to distinguish generated from real data, or measuring *linear* correlations and univariate marginal fidelity for TREND. To highlight this issue, we introduce fully-factorised distributions as baselines, and show how they can perform on par with DGMs under these flawed metrics (Sec. 2).

Second, we propose an embarrassingly simple baseline for TDG that matches diffusion-based SotA models while being one or two orders of magnitude faster (see Fig. 1): *Probabilistic Circuits* (PCs; Vergari et al. 2020, Choi et al. 2020, Darwiche 2003), which can be understood as the hierarchical version of mixture models [McLachlan et al., 2019], and as the generative equivalent of decision trees and forests [Correia et al., 2020, Khosravi et al., 2020]. As such, PCs inherit the ability to seamlessly handle heterogeneous data and scarce sample sizes. Moreover, PCs with tree-like structures have been extensively trained on tabular data in the past [Molina et al., 2018, Vergari et al., 2019b, Watson et al., 2023], especially on binary data and in the form of sparse computational graphs running on the CPU [Zhao et al., 2016, Di Mauro et al., 2017, Dang et al., 2020]. However, modern PCs with tensorised structures [Peharz et al., 2020b,a] and better learning algorithms [Mari et al., 2023, Loconte et al., 2025a] have never been evaluated for TDG. We argue that the TDG community has been missing a strong baseline which, as we show, can easily outperform more recent DGMs, while providing tractable inference.

Our contributions can therefore be summarised as follows. **C1)** We question the effectiveness of standard metrics for evaluating TDG such as C2ST and TREND [Dat, 2025] in Sec. 2, and propose alternatives that better illustrate how the current SotA stands. **C2)** We introduce TABPC as tensorised PCs for TDG in Sec. 3, by showing how it is simple to learn modern PC architectures [Mari et al., 2023, Loconte et al., 2025a, Liu and Van den Broeck, 2021] on heterogeneous data. Finally, **C3)** we show in a rigorous and extensive set of experiments in Sec. 4 that TABPC can *outperform and compete* with current SotA DGMs for TDG under fidelity and utility metrics *in a fraction of the time*.

2 THE SOTA OF TABULAR DATA GENERATION(?)

Before discussing the limitations in commonly-used evaluation protocols for measuring data fidelity, we briefly introduce the current SotA of DGM for TDG. GAN-based approaches started the trend of DGMs for TDG with methods such as CTGAN [Xu et al., 2019], concurrently with VAE approaches such as TVAE [Xu et al., 2019] and HI-VAE [Nazabal et al., 2020, Javaloy et al., 2022]. More recently, diffusion-based approaches such as CODI [Lee et al., 2023], STASY [Kim et al., 2023] and CDTD [Mueller et al., 2025] were introduced.

At the time of writing, the current SotA for DGMs for TDG is given by diffusion-based approaches like TABDIFF [Shi et al., 2025] and TABSYN [Zhang et al., 2024], which provide more sophisticated ways to perform diffusion, but we remark that TDG is an actively growing field [Mueller et al., 2026]. Recently, foundation models such as TABPFN [Hollmann et al., 2023] and TabICL [QU et al., 2025] have gained popularity for discriminative tabular tasks. Extensions of TABPFN[1] provide functionality for the generation of tabular data. To the best of our knowledge, as of yet this has seen only limited benchmarking in one concurrent work, focused primarily on causally-informed generation and not the general task [Tugnoli et al., 2026].

One aspect that sets tabular data apart from modalities such as image or text, is that *there is no clear way to qualitatively assess sample quality*. We cannot simply look at a generated row and say "yes, that looks like a realistic sample". It is therefore crucial to develop meaningful metrics to evaluate synthetic data quality. Recent works [Zhang et al., 2024, Guzmán-Cordero et al., 2025, Shi et al., 2025] use a number of metrics that can be grouped along these axes: *i)* statistical similarity to the original data (*fidelity*); *ii)* performance on downstream tasks, when using synthetic data to train a regressor/classifier and evaluating on real data (*utility* of synthetic data); and *iii)* protection of synthetic data against leaking sensitive information (*privacy*) [Stoian et al., 2025]. We now focus on fidelity metrics such as TREND and C2ST, as they are arguably the ones most prominently used to assess the SotA for TDG, and privacy metrics have already been critically analysed in Yao et al. 2025. We will discuss the evaluation of utility metrics in App. E.4.

2.1 PERFECT FIDELITY?

SotA DGMs such as TABDIFF [Shi et al., 2025] and TABSYN [Zhang et al., 2024] have been reported to achieve *almost perfect scores* in TREND and C2ST, i.e., values close to 1, as we also show in Tabs 5 and 7, thus suggesting almost-perfect generation. However, as we show next, this does not mean that data they generate is really indistinguishable from original data (C2ST), nor that it truly recovers feature dependencies (TREND). To this end, we consider the simplest generative model: a *fully factorised* (FF) model, assuming

[1] https://github.com/PriorLabs/tabpfn-extensions

each feature (column) to be independent, defined as

$$p_{\text{FF}}(\boldsymbol{x}) = \prod_{i=1}^{D} p_i(x_i) \tag{FF}$$

where $\boldsymbol{x} \coloneqq \{x_1, \ldots, x_D\}$ denote the set of random variables representing the table features, and each p_i is modelled as a Gaussian distribution for continuous features and a categorical distribution for discrete features. This yields a parsimonious model, with only $\mathcal{O}(D)$ trainable parameters. Maximum likelihood estimation (MLE) training is almost instantaneous as we just need to sweep over the training data once to compute sufficient statistics (after the standard data preprocessing done for DGMs detailed in App. D.2).

Clearly, the FF model is fundamentally limited, since *it cannot generate correlated features by design*, and one would expect its fidelity as measured by TREND and C2ST to be low, except for trivial datasets. However, this turns out not to be the case. On TREND, the FF model is able to achieve competitive or *occasionally superior* scores to SotA diffusion-based approaches. For example, on the Diabetes dataset (further details in Sec. 4 and App. D.1), TABDIFF scores 0.9711 and TABSYN 0.9593, while FF scores 0.9686, nearing the perfect score of 1. FF beating a diffusion-based model in a metric supposedly measuring dependencies is clearly problematic. Similarly, FF achieves a *perfect* C2ST utility score of 1 on the Diabetes dataset, in comparison with 0.9591 for TABDIFF and 0.6476 for TABSYN. This problem is exacerbated by the fact that these two metrics are used in several recent works to evaluate TDG of SotA models [Zhang et al., 2024, Shi et al., 2025, Guzmán-Cordero et al., 2025]. Next, we investigate how and why exactly FF is able to 'fool' these metrics.

2.2 OFF TREND: MODELLING DEPENDENCIES

To understand why a simple FF model is able to match SotA DGMs for TREND, we show that this metric is sensitive to the quality of the model's univariate marginals. We later propose an alternative that resolves this issue. TREND computes the average similarity of correlations between feature pairs in both real and synthetic data. Specifically, for each pair of continuous features x_i, x_j, it computes

$$s_{\text{corr}}(x_i, x_j) = 1 - 0.5|\rho_R(x_i, x_j) - \rho_S(x_i, x_j)|, \tag{1}$$

where $\rho_R(x_i, x_j)$ is the Pearson correlation between columns x_i and x_j in the real data, and similarly for ρ_S in the synthetic data. For each pair of categorical features x_i, x_j, it instead computes the total variation distance between the real and synthetic distributions, given by

$$s_{\text{contingency}}(x_i, x_j) = 1 - \frac{1}{2} \sum_{\alpha \in x_i} \sum_{\beta \in x_j} |R_{\alpha,\beta} - S_{\alpha,\beta}|, \tag{2}$$

where $R_{\alpha,\beta}$ denotes the empirical probability of jointly observing $x_i = \alpha$, $x_j = \beta$ in the real data, and similarly for $S_{\alpha,\beta}$ in the synthetic data. For pairs of mixed continuous and categorical features, it first discretises the numerical column and then computes the contingency similarity. The overall TREND score is then given by the average of the scores of all unique column pairs.

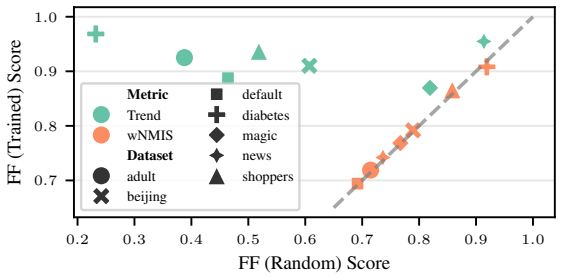


Figure 2: **wNMIS is invariant to the quality of the univariate marginals while TREND is not**, as shown by the fact that it assigns almost identical scores to FF models trained via MLE and to FF models with randomly initialised parameters across all datasets (denoted by marker shape), lying on the identity denoted as the grey dashed line. In contrast, TREND says that a trained FF model captures bivariate dependencies well, and an untrained model does not.

As one can clearly see, TREND **i)** only measures linear correlation among continuous features and thus fails to capture more complex dependencies. More crucially, however, **ii)** it can be fooled by the quality of univariate marginals. In fact, a FF model with randomised parameters scores less on TREND than a FF model learned by MLE (see Fig. 2), while both should ideally score the same values for a measure of pairwise dependencies, which both models cannot capture. As a result, TREND captures the marginal quality more than pairwise dependencies.[2] Instead, an ideal metric for bivariate dependencies should be invariant to marginal transformations, on top of capturing rich non-linear dependencies.

To overcome the limitations **(i-ii)** of TREND, we follow Xu and Veeramachaneni 2018, Yang et al. 2024 who proposed to use mutual information (MI) to measure bivariate dependencies. Specifically, we propose a *weighted normalised mutual information (NMI) similarity (*wNMIS*)* that weights NMI by emphasising which feature pairs have high NMI, and penalises those with low NMI. For features x and y, the NMI [Witten et al., 2011] is defined as

$$\text{NMI}(x, y) = 2\text{I}(x, y)/(\text{H}(x) + \text{H}(y)) \in [0, 1], \tag{3}$$

where I and H refer to the mutual information and entropy, respectively. Specifically, we compute wNMIS as follows:

$$\sum\nolimits_{\{x_i, x_j\,:\,i<j\}} \widetilde{w}(x_i, x_j)\text{NMIS}(x_i, x_j) \in [0, 1], \tag{4}$$

[2] SHAPE is the fidelity metric used to measure univariate marginal quality [Dat, 2025], and as expected a MLE-trained FF model scores almost perfectly on it as well (see Tab. 4 in our experiments).

where x_i, x_j are two different columns of the dataset, and

$$\begin{aligned}\text{NMIS}(x_i, x_j) &\coloneqq 1 - |\text{NMI}_R(x_i, x_j) - \text{NMI}_S(x_i, x_j)| \\ w(x_i, x_j) &\coloneqq |\text{NMI}_R(x_i, x_j) + \text{NMI}_S(x_i, x_j)|, \\ \widetilde{w}(x_i, x_j) &\coloneqq w(x_i, x_j) / \sum_{i<j} w(x_i, x_j)\end{aligned}$$

where NMI_R denotes the NMI between features in the real dataset, and NMI_S in the synthetic one. We can estimate wNMIS as $\widehat{\text{wNMIS}}$ by replacing I and H in Eq. (3) with their empirical estimators after discretising continuous features. We provide an ablation study on the effect of different discretisation schemes on wNMIS score in App. H.4. This score is measured from 0 (worst) to 1 (best), thereby matching the semantics of TREND. Crucially, it does not suffer from issue **(ii)** of TREND, as discussed next.

Its invariance to univariate marginal quality is reflected empirically in our previous experiment with a randomised FF model. Fig. 2 shows the wNMIS scores are the same, up to numerical precision, for both FF models. Note that the wNMIS score of a FF model is not zero but highlights how complex the bivariate interactions in a dataset are, in an analogous way in which the MI of a feature with itself is its entropy.

2.3 BETTER CLASSIFIERS, LOWER FIDELITY

The next fidelity metric we consider, C2ST [Lopez-Paz and Oquab, 2017], trains a classifier to distinguish real from synthetic data, with increasing classifier performance implying lower data quality. This is based on the idea that synthetic data should be indistinguishable from real data. C2ST is defined as follows. Given real and synthetic datasets $\mathcal{D}_R$ and $\mathcal{D}_S$ (respectively), with $|\mathcal{D}_R| = |\mathcal{D}_S|$, one forms balanced training ($\mathcal{D}^{(\text{train})}$) and test ($\mathcal{D}^{(\text{test})}$) sets out of concatenating $\mathcal{D}_R$ and $\mathcal{D}_S$, then trains a binary classifier c to determine synthetic samples on $\mathcal{D}^{(\text{train})}$, and finally computes its area under the receiver operating curve (AUROC) [Hanley and McNeil, 1982] on $\mathcal{D}^{(\text{test})}$. The overall C2ST score, denoted as $\text{C2ST}(c, \mathcal{D}_R, \mathcal{D}_S)$, is then given by

$$1 - (2 \cdot \max(\text{AUROC}(c, \mathcal{D}^{(\text{test})}), 0.5) - 1). \quad \text{(C2ST)}$$

FF models are able to achieve high C2ST scores because it is standard to use Logistic Regression (LR) as the classifier, c, as seen in recent SotA work [Shi et al., 2025, Zhang et al., 2024, Guzmán-Cordero et al., 2025] and the library `SDMetrics` [Dat, 2025]. Intuitively, these unusually high scores may happen because a LR is not powerful enough as a classifier. This intuition is formalised in the theorem below.

Theorem 2.1. *Let $\mathcal{D}_R$ with $|\mathcal{D}_R| = n$ be a real dataset, and let p be an FF model trained by MLE on this dataset. Then*

$$\lim_{n\to\infty} \mathbb{E}_{\mathcal{D}_S}[\text{C2ST}(LR, \mathcal{D}_R, \mathcal{D}_S)] = 1,$$

where $\mathcal{D}_S$ is an i.i.d. sample of n items drawn from p.

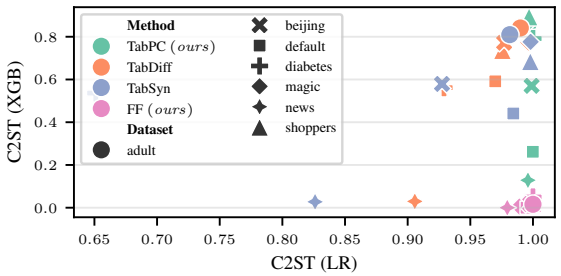


Figure 3: **C2ST with XGBoost offers a much clearer stratification of model performance than with LR** across all datasets, and is able to correctly separate out the performance of the trivial FF model. Details in Sec. 2.2.

We prove this result in App. A by noting that any distribution that recovers the data univariate marginals will maximise C2ST implemented with a LR classifier. This, in addition to our results on TREND being insufficient in capturing bivariate dependencies, begs the question: are SotA DGMs just recovering the data marginals? The short answer is no (see Sec. 4), but a longer version reveals that under more robust metrics, these models' performance is far from optimal. We now look into estimating C2ST more robustly.

Some prior [Zein and Urvoy, 2022] and concurrent [Kindji et al., 2025, Mueller et al., 2026] works have used more powerful boosted tree classifiers such as XGBoost [Chen and Guestrin, 2016] instead of LR. However, they do not adopt the same evaluation protocol as SotA DGMs and thus do not observe the issue of using LR in the context of evaluating DGMs. We therefore emphasise here that ***logistic regression is not a suitable choice of classifier for C2ST and we want to popularise the use of XGBoost for C2ST in TDG benchmarks***.

In fact, using an insufficiently powerful classifier for C2ST has significantly inflated our perception of how well we are creating 'indistinguishable' data. For example, TABDIFF's C2ST on the Adult dataset goes from an almost perfect 0.9899 using LR to 0.8409 using XGBoost, while a FF model goes from 1.0 to 0.0160. More concerningly, TABDIFF also goes from 0.9057 to a dismal 0.0296 on the News dataset. It is clear then that we still have some way to go, as supported by Fig. 3 and complete results in Tabs 7 and 8.

The FF model has served as a cheap and simple diagnostic benchmark, uncovering flaws in existing fidelity metrics by achieving undeservedly high scores. In the next section, we start from the FF model to devise an expressive and efficient approach to TDG, that retains simplicity and efficiency while delivering competitive performance on stronger metrics such as wNMIS and C2ST with XGBoost.

3 FROM FF MODELS TO DEEP TABPC

3.1 FROM FF TO SHALLOW MIXTURES...

A natural way to reintroduce dependencies among features, starting from a FF model, is to combine several of them into a *shallow mixture* (SM) [McLachlan et al., 2019], defined as

$$p_{\text{SM}}(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \prod_{j=1}^{D} p_{i,j}(x_j) \qquad \text{(SM)}$$

where $w_i \in \mathbb{R}_{>0}$, $\sum_i w_i = 1$ are additional learnable *mixture weights* determining the contributions of the K mixture components. The base distributions, $p_{i,j}$, can be Gaussians or categoricals, depending on the feature type. Compared to FF models, the number of parameters will linearly increase to $\mathcal{O}(KD)$, and MLE training can be done via expectation maximisation (EM) or stochastic gradient ascent, as SM models introduce a categorical latent variable with K states that is marginalised out to compute $p_{\text{SM}}(\boldsymbol{x})$ [Peharz et al., 2017]. This latent variable interpretation also enables efficient exact sampling: one first samples the latent variable state proportionally to the mixture weights, and then samples from the selected component directly [Peharz et al., 2017].

Despite introducing a single latent variable, SM models are universal density approximators in the limit of infinite mixture components [McLachlan et al., 2019]. That is, by increasing K we can increase model expressiveness. For example, one can see an increase in C2ST with XGB, from 0.0160 on Adult as scored by a FF model to 0.7727 for a SM with $K = 20,000$. This is remarkable for such simple baselines, as TabDiff, having $7.87\times$ more parameters scores 0.8554 (see Sec. 2.3). We study the results of SM models in greater detail in App. E.3. Next, we discuss how to use these ideas to build *hierarchical and overparameterised mixture models* in the form of PCs. As per the deep learning recipe, we will increase expressiveness by building *deeper*, and *not wider*, mixture models [Martens and Medabalimi, 2014, Choi et al., 2020].

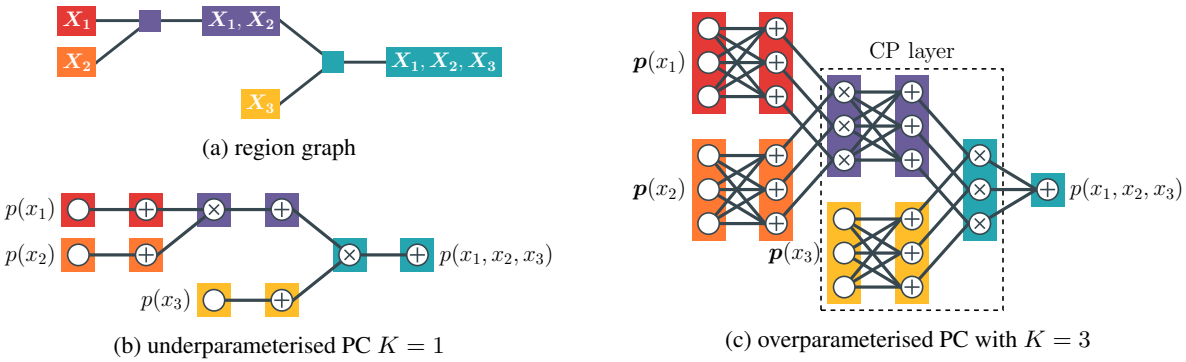
3.2 ...AND TO DEEP PROBABILISTIC CIRCUITS

Before detailing our 'recipe' for building and learning of a PC for tabular data (TABPC), we briefly review PCs. PCs provide a framework to model hierarchical mixture models as deep computational graphs over which one can systematically trade-off expressiveness for tractability by governing the number of parameters in them and certifying that certain structural properties of the graph are met [Vergari et al., 2019a]. Furthermore, many other classical tractable probabilistic models such as HMMs and trees [Correia et al., 2020, Watson et al., 2023], as well as tensor factorisations, are instances of PCs [Choi et al., 2020, Correia et al., 2020, Khosravi et al., 2020, Loconte et al., 2025a].

Formally, a *circuit* [Darwiche, 2003, Choi et al., 2020, Vergari et al., 2021] c is a parameterised directed acyclic computational graph over variables $\boldsymbol{x}$ encoding a function $c(\boldsymbol{x})$. A circuit comprises three kinds of computational units: *input*, *product*, and *sum*. Each sum or product unit n receives the outputs of other units as inputs, denoted as the set $\mathsf{in}(n)$. Each n encodes a function c_n defined as: (i) a tractable function $f_n(\mathsf{sc}(n); \theta)$ if n is an *input unit* with parameters θ, defined over variables $\mathsf{sc}(n) \subseteq \boldsymbol{x}$, called its *scope*; (ii) $\prod_{j \in \mathsf{in}(n)} c_j(\mathsf{sc}(j))$ if n is a *product unit*; and (iii) $\sum_{j \in \mathsf{in}(n)} w_{n,j} c_j(\mathsf{sc}(j))$ if n is a *sum unit*, where each $w_{n,j} \in \mathbb{R}$ is a parameter of n. The scope of a sum or product unit is the union of the scopes of its inputs, i.e. $\mathsf{sc}(n) = \bigcup_{j \in \mathsf{in}(n)} \mathsf{sc}(j)$. Then, a *probabilistic circuit* (PC) is a circuit c encoding a non-negative function, i.e., $c(\boldsymbol{x}) \geq 0$ for any $\boldsymbol{x}$, thus encoding a (possibly unnormalised) probability distribution $p(\boldsymbol{x}) \propto c(\boldsymbol{x})$. In a PC, input units can model probability densities (e.g., Gaussians) or masses (e.g., categorical distributions) [Molina et al., 2018]. Note that both the FF model (in Eq. (FF)) and SM model (in Eq. (SM)) are special cases of PCs, as they can be represented as simple computational graphs in this language, as shown in Fig. 8.

A PC c supports the tractable marginalisation of any subset of its variables, and hence also renormalisation, in a single forward step [Choi et al., 2020] if (i) its input functions f_n can be integrated tractably, and (ii) it is *smooth* and *decomposable* [Darwiche and Marquis, 2002]. A circuit is *smooth* if for every sum unit n, all of its input units depend on the same variables, i.e. $\forall i, j \in \mathsf{in}(n) \colon \mathsf{sc}(i) = \mathsf{sc}(j)$. A circuit is *decomposable* if the inputs of every product unit n depend on disjoint sets of variables, i.e. $\forall i, j \in \mathsf{in}(n)\ i \neq j \colon \mathsf{sc}(i) \cap \mathsf{sc}(j) = \emptyset$. Throughout this work, we will assume all our PCs are smooth and decomposable by construction (see Sec. 3.3), which allows the interpretation of PCs as hierarchical latent variable models [Peharz et al., 2017, Gala et al., 2024a,b].

Beyond just sampling for tabular data. Therefore, smoothness and decomposability also enable exact sampling in time linear in the circuit size, i.e., the number of edges in it, which can be done via ancestral sampling by recursively generalising the way in which SM models are sampled, as detailed in App. B.2. More crucially, smoothness and decomposability will allow TABPC to perform certain inference tasks that are out of the reach of all the other SotA DGMs. First, we can compute exact normalised likelihood scores, which we can use for model selection in our experiments (Sec. 4). Second, one can handle missing values 'on the fly' both at inference and training time. Therefore, TABPC can also perform exact conditional sampling for *any* subset of provided evidence configuration, *without* retraining. We evaluate these additional capabilities of TABPC in Sec. 4.2.



(a) region graph

(b) underparameterised PC $K = 1$

(c) overparameterised PC with $K = 3$

Figure 4: **Constructing TABPC requires three choices: the region graph (RG), level of overparameterisation, and type of sum-product layers.** Fig. 4a shows a tree-shaped RG over three variables. This acts as a template from which we construct the simple circuit shown in Fig. 4b. To increase expressivity, we *overparameterise* the circuit by populating it with K units organised in *layers*, as seen in Fig. 4c for $K = 3$. Finally, the choice of sum-product layer dictates how we connect units across layers. TABPC uses CP sum-product layers, also shown in Fig. 4c, which are described in detail below.

3.3 BUILDING TABPC

Several past works have learned PCs for tabular data, with the majority focussing on learning their structure, i.e., the edges in their graphs [Vergari et al., 2015, Molina et al., 2018, Vergari et al., 2019b, Di Mauro et al., 2017, Watson et al., 2023] and parameters [Zhao et al., 2016] but assuming their computational graphs to be trees instead of directed acyclic graphs (DAGs). More recent works shifted to learning PCs over image and text modalities [Liu et al., 2023a, Gala et al., 2024a, Zhang et al., 2025], prescribing the structure of PCs to be a fixed but *overparameterised* DAG, and learning only the parameters [Peharz et al., 2020b,a, Liu et al., 2024, Loconte et al., 2025a, Suresh et al., 2025]. As such, modern tricks to scale building and learning PCs have not been explored for TDG, yet.

In this section, we detail the construction of TABPC, which uses modern recipes for overparameterised DAG PCs. Specifically, we piggyback on the "Lego block" approach to construct smooth and decomposable tensorised PCs [Loconte et al., 2025a], which is summarised in the following 'recipe': (i) the **choice of region graph (RG)**, (ii) the **level of overparameterisation**, and (iii) the **choice of sum-product layers**. We remark that few adaptions must be made to unlock TDG for PCs; the flexibility of the framework enables their use almost 'out of the box', highlighting how little is needed to yield a strong baseline that resets the SotA for TDG.

(i) Region Graph (RG): Given the set of features of a table, a RG is a template to build PCs that are smooth and decomposable by design [Peharz et al., 2020b]. A RG is a bipartite DAG comprising regions, i.e., sets of features, and partitions, describing a hierarchical partitioning of these features. Each region and partition in a RG will serve as the basis of a *layer* in a PC. Fig. 4a illustrates a tree RG over three features and App. B.1 formalises its construction.

Several approaches to RG construction have been proposed in the literature [Loconte et al., 2025a]. For TABPC, we use a method which builds a RG from a Chow-Liu tree [Chow and Liu, 1968] learned on the training data [Dang et al., 2021, Liu and Van den Broeck, 2021]. This is done by computing the MI between feature pairs and iteratively adding the maximum MI feature pair to the tree. The constructed tree can then be compiled directly into a RG; see Loconte et al. 2025a and App. B.1 for more details. We then build our PC according to the RG, associating input layers to leaf regions, and product and sum layers to partitions and inner regions in the RG. Each layer is a logical abstraction that can correspond to a set of units in the circuit.

(ii) Overparameterisation: In principle, one can use a single unit per layer. This would build a simple circuit obeying our desired structural constraints. Fig. 4 provides an example of this process. Alternatively, one can *overparameterise* the layered PC: in the same way we increase the number of mixture components K in a SM model, we populate each node of the RG with not just one, but K sum, input, and product units. This allows for computations to be executed in parallel, thereby increasing efficiency. In practice, these layers are computed as tensors, i.e. are *tensorised*. More crucially, deep overparameterised circuits are more expressive efficient than shallow mixtures, i.e., using K units per layer yields the expressivity of a mixture with $K^{N/2}$ components, where N is the depth of the PC [Choi et al., 2020]. In our experiments, we use K on the order of 10^3 units per layer (specific values per dataset can be found in App. D.5). However, this will not be a drawback when it comes to speed and performance, as shown in Sec. 4.

(iii) Sum-Product Layer: After we decide on the number of units K, we need to connect them across layers. For TABPC, we use *CP sum-product layers*, described in Loconte et al. 2025a. The CP sum-product layer computation is defined as follows: if the scope of CP-layer $\boldsymbol{\ell}$ is $\boldsymbol{y} \subseteq \boldsymbol{x}$, and we assume

it has two input layers ℓ_1, ℓ_2 with disjoint scopes $\boldsymbol{y}_1, \boldsymbol{y}_2$, with $\boldsymbol{y}_1 \cup \boldsymbol{y}_2 = \boldsymbol{y}$, then the CP-layer $\boldsymbol{\ell}$ computes

$$\boldsymbol{\ell}(\boldsymbol{y}) = \left(\boldsymbol{W}^{(1)}\boldsymbol{\ell}_1(\boldsymbol{y}_1)\right) \odot \left(\boldsymbol{W}^{(2)}\boldsymbol{\ell}_2(\boldsymbol{y}_2)\right) \tag{5}$$

for some weight matrices $\boldsymbol{W}^{(i)} \in \mathbb{R}^{K\times K}$, $i \in \{1, 2\}$ and where $\odot$ denotes the Hadamard (element-wise) product.

Locally, we can view this as a tensor of K distributions, each of which factorises into two shallow mixture models over their respective variables $\boldsymbol{y}_1, \boldsymbol{y}_2$. Here, smoothness gives us the interpretation of a mixture model, since the scopes of all mixture components are the same, and decomposability gives us the factorisation, since the scopes of both sum layers are disjoint. By stacking these on top of each other according to the RG, as can be seen in Fig. 4, we build a deep, hierarchical mixture model—i.e., a PC. App. B.3 details how the evaluation of these tensorised TABPC can be sped up further on the GPU. Once built, one can train PCs by MLE either by stochastic gradient ascent, or by EM [Peharz et al., 2017, 2020a]. In our experiments, we train TABPC using the RAdam optimiser [Liu et al., 2019].

In summary, we use TABPC to denote our specific implementation for TDG: an overparameterised PC built from a CLT RG and using CP layers.

4 RE-EVALUATING SOTA FOR TDG

We now evaluate TABPC against SotA DGMs models on a range of popular benchmarks for TDG, starting from the experimental protocol of Shi et al. 2025, Zhang et al. 2024, Guzmán-Cordero et al. 2025, but updating it after our observations in Sec. 2. We aim to answer the following questions: **Q1)** is TABPC competitive in terms of fidelity, utility and time? **Q2)** how much does the SotA change when we use WNMIS and C2ST with XGBoost for assessing fidelity? **Q3)** can we effectively harness tractable inference with TABPC for conditional sampling and model selection? **Q4)** how do TABPC and existing SotA DGMs compare against the generative extension of foundation models such as TABPFN?

Datasets: We evaluate on the commonly used UCI datasets with heterogeneous features: *Adult*, *Beijing*, *Default*, *Diabetes*, *Magic*, *News*, and *Shoppers*. App. D.1 details their statistics and App. D.2 describes their pre-processing.

Baselines: We follow Zhang et al. 2024, Shi et al. 2025 and compare against SotA DGMs for TDG, including CTGAN, TVAE, GREAT [Borisov et al., 2023] and STASY, CODI, TABSYN, and TABDIFF. See Sec. 2 and App. D.4 for details. Additionally, to answer Q4 we compare against the data generation extension of TABPFN v3 (App. D.4.1).

Metrics: We evaluate against the following metrics (organised by category). Each is described in greater technical detail in App. D.3. (i) **Fidelity**: SHAPE, TREND, and C2ST (LR) (implemented using `SDMetrics` [Dat, 2025]), and α-*precision* and β-*recall* [Alaa et al., 2022] (implemented using `Synthcity` [Qian et al., 2023]). We additionally evaluate against our promoted metrics C2ST (XGB) and WNMIS. Due to space constraints we push to App. E.4 the discussion about (ii) **Utility**: *machine learning efficacy* [Xu et al., 2019] (re-using the implementation of Shi et al. 2025). We provide our results, as well as a further discussion of interpreting and evaluating performance on this metric. (iii) **Privacy**: App. E.5 discusses our results on distance-based privacy metrics, such as the commonly used *distance to closest record (DCR)*. However, we remark that DCR has been strongly criticised in Yao et al. 2025 as "flawed by design", and hence we do not focus in detail on these results.

Evaluation protocol: Differently from Zhang et al. 2024, Shi et al. 2025, who train a single model per dataset and use it to generate several data samples, we rerun each baseline with five seeds, and report averaged metrics with proper standard deviations. This is with the exception of TABPFN, which does not require retraining or fine-tuning, and hence we compute its metrics averaged over five generations (where possible). Apart from reporting full results and average ranks in App. E.1, we also compute critical difference diagrams (CDDs), which provide an indication of whether the performance of two methods are statistically different from each other. App. E.2 collects all CDDs.

4.1 Q1 + Q2: FIDELITY METRICS & TIME

SHAPE is designed to measure how well synthetic data models the univariate marginals. Tab. 4 shows that TABPC outperforms all previous SotA models in six out of the seven datasets, consistently achieving scores above 0.99, close to the maximum value of 1. The saturation of this metric suggests that all recent SotA models accurately recover univariate marginals, and scoring well on SHAPE is necessary but not sufficient to generate realistic data.

TREND and WNMIS: As we have noted, the TREND metric is problematic, and so we instead consider WNMIS scores to measure bivariate dependency quality (for reference, TREND results are reported in Tab. 5, and WNMIS results in Tab. 6. In terms of WNMIS, TABPC outperforms existing SotA models in four out of seven datasets, and achieves scores greater than 0.99 in five datasets. Again, metric saturation suggests that SotA models already capture low-order correlations relatively effectively, and investigating higher-order information would be required for better stratification of model performance.

C2ST (LR / XGB): In comparison to SHAPE and TREND / WNMIS, C2ST considers samples from the full joint, and not just univariate or pair-wise statistics. However, as we discussed in Sec. 2.3, C2ST (LR) is flawed, and hence we instead consider the results using XGBoost as the classifier

(LR results are reported in Tab. 7 for reference, and XGB results in Tab. 8). In terms of C2ST (XGB), TABPC offers competitive performance, outperforming the diffusion-based SotA in four out of the seven tested datasets. We note that all generally high-performing models struggle on the News dataset, which has a large number of numerical features (i.e. 46 out of 48 features are numerical) and skewed distributions, and remark again that under C2ST (LR) this struggle was not visible.

α-PRECISION and β-RECALL: These metrics extend the notion of precision and recall to generative models [Alaa et al., 2022]. α-PRECISION measures synthetic data 'realism', whereas β-RECALL measures synthetic data 'diversity'. Full results are in Tab. 9 for α-PRECISION and Tab. 10 for β-RECALL. The CDDs in Fig. 12 show that TABPC is in the top clique for both α-PRECISION and β-RECALL along with TABDIFF and TABSYN (and a few other methods for α-PRECISION), suggesting that TABPC offers competitive fidelity and diversity of samples to these methods. Note that backwards compatibility of these results with specific values reported by certain prior works is problematic; see App. I.

Time and discussion: TABPC is able to deliver the performance described above one or two orders of magnitude faster than other DMGs in terms of training time; see Fig. 1 and Tab. 12. We remark that our implementation can be further sped up by leveraging recent advancements in scaling PCs over GPUs [Liu et al., 2024, Zhang et al., 2025]. Moreover, we note that TABPC provides the best ranked dataset sampling time of the current SotA TDG methods, in particular beating TABDIFF and TABSYN in six out of the seven datasets; see Tab. 13.

4.2 Q3: BEYOND UNCONDITIONAL SAMPLING

As discussed in Sec. 3, TABPC allows for the efficient computation of exact likelihoods and arbitrary marginals. We now evaluate these capabilities for model selection and conditional sample generation.

Model selection via likelihood: We investigate if there is a correlation between the fidelity metrics discussed above with the validation likelihood for TABPC. For high dimensional data such as images, high likelihood does not always corresponding to good sample quality [Theis et al., 2015, Lang et al., 2022, Braun et al., 2025]. As we show next, this is not the case for tabular data. Note that this type of analysis is only possible with PCs, as all previous SotA models do not allow for exact likelihood computation.

Concretely, we compute the validation bits-per-dimension (BPD, see App. F.1) and downstream sample quality, as measured by C2ST (XGB) for the different TABPC models we trained with various hyperparameters—specifically, number of sum and input units (K), learning rate, and batch size. Fig. 5 shows that decreasing BPD (corresponding to an increased log-likelihood) strongly correlates with increasing sample C2ST (XGB) (and similar plots for all datasets can be found in App. F). As PCs struggle to generate high-quality image samples while delivering low bpds [Lang et al., 2022, Braun et al., 2025], we highlight that tabular data is a better modality for this kind of generative model.

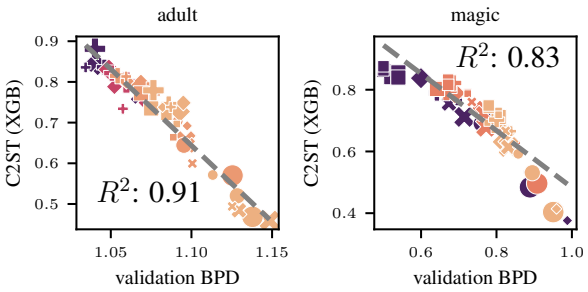


Figure 5: **Validation bits-per-dimension (BPD) provides a strong signal of downstream sample quality** as measured by C2ST (XGB), as displayed here for the Adult *(left)* and Magic *(right)* datasets (full results in App. F.2). Each point represents a different hyperparameter configuration for TABPC, across number of sum and input units (colour), batch size (marker size), and learning rate (marker style). The dashed grey line is a Huber regression fit.

Exact and efficient conditional sampling: As discussed in Sec. 3.2, circuits can exactly condition on any arbitrary feature subset. This enables us to evaluate the fidelity of conditionally generated data in a systematic way. Note that this is in stark contrast to other DGMs such as diffusion which cannot marginalise exactly and can only retrieve conditional sampling with either heuristics involving training over conditioning masks [Song et al., 2020] or more sophisticated MCMC schemes [Wedenig and Peharz, 2025].

Fig. 6 reports C2ST (XGB) scores when conditioning on different percentages of randomly selected features on the test set. Specifically, 0% of conditioning (left) corresponds to unconditional generation, while observing 100% (right) of the dataset simply means copying it, hence maximum performance. We can see how TABPC performance quickly increases and never drops (thanks to exact conditioning) and it is able to generate completions which almost perfectly fool the XGBoost classifier with already 50% observations, hence displaying a high degree of realism. Note how a simple baseline such as mean imputation (or mode imputation for categorical features) grows much slower. Fig. 17 shows the same trend for wNMIS. We remark that we condition on a different set of features for each data row, this would make it infeasible to perform heuristic imputation through optimisation [Ho and Salimans, 2022] with diffusion models.

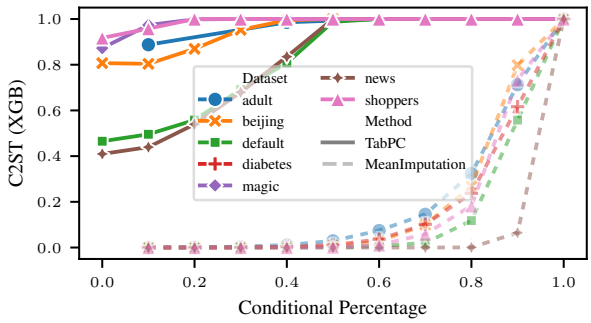


Figure 6: **TabPC produces high fidelity conditional samples**, as shown by its ability to produce datasets with a high C2ST (XGB) score when **seamlessly performing exact conditioning** on various percentages of the unseen test set. Solid lines correspond to results from datasets generated by TabPC, dashed translucent lines correspond to mean imputation (or mode imputation for categorical features).

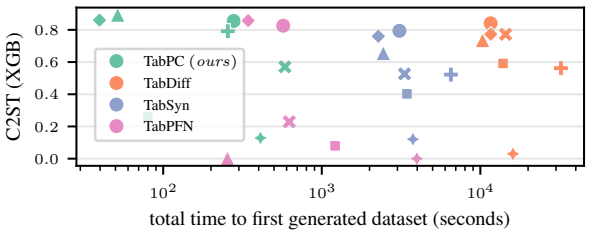


Figure 7: **TabPC models can train *and* generate one dataset in less time than it takes to generate one dataset from TabPFN,** while providing superior fidelity under our metrics. Though producing similar fidelity data to TabPC, TabDiff and TabSyn come at the cost of a longer total time. For TabPFN we just plot the time to generate one dataset.

4.3 Q4: Deep Generative Models vs TabPFN

Given the recent promising results of tabular foundation models such as TabPFN on discriminative tasks (e.g. as shown by Erickson et al. 2026), this begs the question: how well can they be extended to generative tasks, and how do they compare to DGMs designed for this task? We focus on TabPFN as one of the current most popular models, and because of its community extensions for unsupervised tasks.

Notably, TabPFN has been extended to provide functionality for synthetic data generation. It does this by sampling features autoregressively following some feature ordering. For robustness, the extension generates according to multiple permutations of the feature ordering, and then averages over these generations. See App. D.4.1 for specific implementation details.

Full results for TabPFN can again be found in App. E.1. In particular, we focus here on its Shape, C2ST (XGB), and wNMIS scores. TabPFN's scores on Shape place it somewhat below the level of the diffusion and PC-based SotA in capturing univariate marginals, falling somewhere between TabSyn and STaSy. For wNMIS, TabPFN again offers performance somewhere between the current SotA and STaSy, suggesting that it is not quite as good at capturing pairwise dependencies as SotA models. In terms of C2ST (XGB), TabPFN offers varying performance. For example, on Magic, TabPFN scores similarly to TabPC (the top-ranking method for this dataset), whereas on News and Shoppers, the score is almost zero. We conclude that the quality of the joint distribution captured by these samples varies greatly per dataset. Also, TabPFN has an average rank above TabSyn and TabPC for β-Recall, and generally scores well for this metric. This suggests that TabPFN is able to generate a comparably diverse range of samples. And, while the utility metric MLE is not a main focus of this work, it also performs well under this; see App. E.4.

We also note that the dataset sampling time is significantly higher for TabPFN than all other methods. See Fig. 7 for comparison against SotA DGMs. This all suggests that, while it is interesting that TabPFN can be adapted to handle generative tasks, its performance in generating high-fidelity data is currently somewhat behind the level of bespoke models. All in all, TabPC stands as the fastest alternative that delivers the best fidelity performance overall.

5 Conclusion

In this work, we ***reset the current SotA for TDG***, highlighting how there is clearly still a long way to go in generating realistic synthetic data. By highlighting some of the flaws in how two fidelity metrics (Trend and C2ST) are currently used, we have determined that ***our progress in fidelity is not as advanced as we may have believed***. We then provided empirical and theoretical evidence to ***support the widespread use of more robust metrics such as* wNMIS *and* C2ST *(XGB)***. Moreover, we introduced the simplest baseline that sets ***a new standard in trading-off computation for downstream data quality***, TabPC, which opens up future venues in TDG such as efficient conditional generation without retraining. We also note that our TabPC implementation can benefit from further advancements from the circuit literature such as introducing negative parameters [Loconte et al., 2024, 2025b, 2026], and continuous latent variables [Correia et al., 2023, Gala et al., 2024a,b], which are expected to boost expressiveness further with a small computational overhead. Lastly, we highlight how we should revisit in future work classical PC papers learning the structure of circuits [Molina et al., 2018, Vergari et al., 2019b, Correia et al., 2020] or proposing different objectives for parameter learning [Watson et al., 2023], noting that they could be extended to our overparameterized setting.

Author Contributions

DS and AV conceived the original idea and they later discussed it with DP, AJ, HG, SS and LB. DS provided the first implementation of TABPC, ran preliminary experiments for hyperparameter selection and proposed the WNMI-based evaluation metric. DP proved Thm. 2.1 with help from HG and AV, provided the final implementation and hyperparameter selection used in the experiments, and is responsible for all tables and plots, with the exception of Fig. 4, later improved by AJ. AJ implemented conditional sampling, with help from DS. HG proposed to compare bpds and downstream performance. DP led the writing of the paper, with help from DS, AJ, HG and AV. AV supervised all phases of the project and provided feedback throughout.

Acknowledgements

We would like to acknowledge the april lab for its support and feedback, in particular Charles Bricout. Furthermore, we would like to thank Eleonora Giunchiglia for her advice and early feedback, and Chris Williams for an insightful discussion about earlier VAE-based approaches to TDG and association measures for the 2×2 contingency table. Also, we would like to thank Gennaro Gala for providing a more efficient sampling implementation for PCs which helped to greatly improve the dataset sampling times. DS and SS were funded by the European Union through grant agreement 101218531 - SydAi. HG was supported by the Royal Academy of Engineering under the Research Fellowship programme. DP, AJ and AV are supported by the "UNREAL: Unified Reasoning Layer for Trustworthy ML" project (EP/Y023838/1) selected by the ERC and funded by UKRI EPSRC.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, volume 35, pages 29944–29959. Curran Associates, Inc., 2022.

Ahmed Alaa, Boris Van Breugel, Evgeny S Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International Conference on Machine Learning*, pages 290–306. PMLR, 2022.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

R. Bock. MAGIC Gamma Telescope. UCI Machine Learning Repository, 2004. DOI: https://doi.org/10.24432/C52C8B.

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

Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations*, 2023.

Steven Braun, Sahil Sidheekh, Antonio Vergari, Martin Mundt, Sriraam Natarajan, and Kristian Kersting. Tractable representation learning with probabilistic circuits. *Transactions on Machine Learning Research*, 2025.

Song Chen. Beijing PM2.5. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5JS49.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic modeling. Technical report, University of California, Los Angeles (UCLA), 2020.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

John Clore, Krzysztof Cios, Jon DeShazo, and Beata Strack. Diabetes 130-US Hospitals for Years 1999-2008. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5230J.

Mark Collier, Alfredo Nazabal, and Chris Williams. VAEs in the presence of missing data. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*, 2020. URL https://openreview.net/forum?id=PnZT5EWoB7.

Alvaro Correia, Robert Peharz, and Cassio P de Campos. Joints in random forests. *Advances in neural information processing systems*, 33:11404–11415, 2020.

Alvaro HC Correia, Gennaro Gala, Erik Quaeghebeur, Cassio De Campos, and Robert Peharz. Continuous mixtures of tractable probabilistic models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7244–7252, 2023.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869, 2023.

Meihua Dang, Antonio Vergari, and Guy Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *International Conference on Probabilistic Graphical Models*, pages 137–148. PMLR, 2020.

Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: A fast and accurate learner of structured-decomposable probabilistic circuits. *The International Journal of Approximate Reasoning (IJAR)*, 140:92–115, 2021.

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50:280–305, 2003.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*, 17:229–264, 2002.

Synthetic Data Metrics. DataCebo, Inc., 10 2025. URL https://docs.sdv.dev/sdmetrics/. Version 0.24.0.

Nicola Di Mauro, Antonio Vergari, Teresa MA Basile, and Floriana Esposito. Fast and accurate density estimation with extremely randomized cutset networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 203–219. Springer, 2017.

A. W. F. Edwards. The measure of association in a 2 × 2 table. *Journal of the Royal Statistical Society. Series A (General)*, 126(1):109–114, 1963. ISSN 00359238. URL http://www.jstor.org/stable/2982448.

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

Kelwin Fernandes, Pedro Vinagre, Paulo Cortez, and Pedro Sernadela. Online News Popularity. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C5NS3V.

Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, June 1962. ISSN 0001-0782. doi: 10.1145/367766.368168. URL https://doi.org/10.1145/367766.368168.

Sébastien Gadat, Ioana Gavra, and Laurent Risser. How to calculate the barycenter of a weighted graph. *Mathematics of Operations Research*, 43(4):1085–1118, 2018. doi: 10.1287/moor.2017.0896. URL https://doi.org/10.1287/moor.2017.0896.

Gennaro Gala, Cassio de Campos, Robert Peharz, Antonio Vergari, and Erik Quaeghebeur. Probabilistic integral circuits. In *AISTATS 2024*, 2024a.

Gennaro Gala, Cassio de Campos, Antonio Vergari, and Erik Quaeghebeur. Scaling continuous latent variable models as probabilistic integral circuits. *arXiv preprint arXiv:2406.06494*, 2024b.

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

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

Andreas Grivas, Lorenzo Loconte, Emile van Krieken, Piotr Nawrot, Yu Zhao, Euan Wielewski, Pasquale Minervini, Edoardo Ponti, and Antonio Vergari. Fast and expressive multi-byte prediction with probabilistic circuits. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=6kCEyw9god.

Andrés Guzmán-Cordero, Floor Eijkelboom, and Jan-Willem van de Meent. Exponential family variational flow matching for tabular data generation. In *Forty-second International Conference on Machine Learning*, 2025.

James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Dirk Hasenclever and Markus Scholz. Comparing measures of association in 2×2 probability tables. 7(1), 2016. doi: 10.2174/1876527001607010020. URL https://openstatisticsandprobabilityjournal.com/VOLUME/7/PAGE/20/FULLTEXT/.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

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

International Conference on Learning Representations 2023, 2023.

Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. Variational autoencoder with arbitrary conditioning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyxtJh0qYm.

Adrián Javaloy, Maryam Meghdadi, and Isabel Valera. Mitigating modality collapse in multimodal vaes via impartial optimization. In *International Conference on Machine Learning*, pages 9938–9964. PMLR, 2022.

Adrián Javaloy, Antonio Vergari, and Isabel Valera. Copa: Comparing the incomparable in multi-objective model evaluation. *arXiv preprint arXiv:2503.14321*, 2025.

Pasha Khosravi, YooJung Choi, Yitao Liang, Guy Van den Broeck, et al. Handling missing data in decision trees: A probabilistic approach. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*, 2020.

Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations*, 2023.

G.Charbel N. Kindji, Lina M. Rojas-Barahona, Elisa Fromont, and Tanguy Urvoy. Tabular data generation models: An in-depth survey and performance benchmarks with extensive tuning. *Neurocomputing*, 658:131655, 2025. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2025.131655. URL https://www.sciencedirect.com/science/article/pii/S0925231225023276.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pages 17564–17579. PMLR, 2023.

Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

Leander Kurscheidt, Paolo Morettin, Roberto Sebastiani, Andrea Passerini, and Antonio Vergari. A probabilistic neuro-symbolic layer for algebraic constraint satisfaction. In *Conference on Uncertainty in Artificial Intelligence*, pages 2431–2471. PMLR, 2025.

Steven Lang, Martin Mundt, Fabrizio Ventola, Robert Peharz, and Kristian Kersting. Elevating perceptual sample quality in pcs through differentiable sampling. In *NeurIPS 2021 Workshop on Pre-registration in Machine Learning*, volume 181 of *Proceedings of Machine Learning Research*, pages 1–25. PMLR, 13 Dec 2022. URL https://proceedings.mlr.press/v181/lang22a.html.

Chaejeong Lee, Jayoung Kim, and Noseong Park. Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *International Conference on Machine Learning*, pages 18940–18956. PMLR, 2023.

Anji Liu and Guy Van den Broeck. Tractable regularization of probabilistic circuits. *Advances in Neural Information Processing Systems*, 34:3558–3570, 2021.

Anji Liu, Honghua Zhang, and Guy Van den Broeck. Scaling up probabilistic circuits by latent variable distillation. In *11th International Conference on Learning Representations (ICLR)*, 2023a.

Anji Liu, Kareem Ahmed, and Guy Van den Broeck. Scaling tractable probabilistic circuits: A systems perspective. *arXiv preprint arXiv:2406.00766*, 2024.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2019. URL http://arxiv.org/abs/1908.03265.

Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. Goggle: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*, 2023b.

Lorenzo Loconte, M. Sladek Aleksanteri, Stefan Mengel, Martin Trapp, Arno Solin, Nicolas Gillis, and Antonio Vergari. Subtractive mixture models via squaring: Representation and learning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Lorenzo Loconte, Antonio Mari, Gennaro Gala, Robert Peharz, Cassio de Campos, Erik Quaeghebeur, Gennaro Vessio, and Antonio Vergari. What is the relationship between tensor factorizations and circuits (and how can we exploit it)? *Transactions on Machine Learning Research (TMLR)*, 2025a. ISSN 2835-8856. Featured Certification.

Lorenzo Loconte, Stefan Mengel, and Antonio Vergari. Sum of squares circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 19077–19085, 2025b.

Lorenzo Loconte, Adrián Javaloy, and Antonio Vergari. How to square tensor networks and circuits without squaring them. In *ICLR*, 2026.

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

Chao Ma, Sebastian Tschiatschek, Konstantina Palla, José Miguel Hernández-Lobato, Sebastian Nowozin,

and Cheng Zhang. EDDI: efficient dynamic discovery of high-value information with partial VAE. *CoRR*, abs/1809.11142, 2018. URL http://arxiv.org/abs/1809.11142.

Chao Ma, Sebastian Tschiatschek, Richard E. Turner, José Miguel Hernández-Lobato, and Cheng Zhang. VAEM: a deep generative model for heterogeneous mixed type data. In *ICML Workshop on the Art of Learning with Missing Values (Artemiss)*, 2020. URL https://openreview.net/forum?id=JZ-6j-siNBj.

Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen – tabular data generation with tabPFN. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023. URL https://openreview.net/forum?id=4MkkNsAEmO.

Antonio Mari, Gennaro Vessio, and Antonio Vergari. Unifying and understanding overparameterized circuit representations via low-rank tensor decompositions. In *The 6th Workshop on Tractable Probabilistic Modeling*, 2023.

James Martens and Venkatesh Medabalimi. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

Frank J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. doi: 10.1080/01621459.1951.10500769. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1951.10500769.

Pierre-Alexandre Mattei and Jes Frellsen. MIWAE: Deep generative modelling and imputation of incomplete data sets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4413–4423. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/mattei19a.html.

Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6(1):355–378, 2019.

Matthew Middlehurst, Ali Ismail-Fawaz, Antoine Guillaume, Christopher Holder, David Guijo-Rubio, Guzal Bulatova, Leonidas Tsaprounis, Lukasz Mentel, Martin Walter, Patrick Schäfer, and Anthony Bagnall. aeon: a python toolkit for learning from time series. *Journal of Machine Learning Research*, 25(289):1–10, 2024. URL http://jmlr.org/papers/v25/23-1444.html.

Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

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

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *Pattern Recognition*, 107:107501, 2020.

Milton Nicolás Plasencia Palacios, Sebastiano Saccani, Gabriele Sgroi, Alexander Boudewijn, and Luca Bortolussi. Contrastive learning-based privacy metrics in tabular synthetic datasets, 2025. URL http://arxiv.org/abs/2502.13833.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2 edition, 2009. ISBN 978-0-511-80316-1 978-0-521-89560-6 978-0-521-74919-0. doi: 10.1017/CBO9780511803161.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(10):2030–2044, October 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2618381. URL https://doi.org/10.1109/TPAMI.2016.2618381.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van Den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: fast and scalable learning of tractable probabilistic circuits. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020a.

Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *35th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 115 of *Proceedings of Machine Learning Research*, pages 334–344. PMLR, 2020b.

Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. Synthcity: facilitating innovative use cases of synthetic data in different data modalities, 2023. URL https://arxiv.org/abs/2301.07573.

Jingang QU, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0VvD1PmNzM.

Jože M. Rožanec, Gašper Petelin, João Costa, Gregor Cerar, Blaž Bertalanič, Marko Guček, Gregor Papa, and Dunja Mladenić. Dealing with zero-inflated data: Achieving state-of-the-art with a two-fold machine learning approach. *Engineering Applications of Artificial Intelligence*, 149:110339, 2025. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2025.110339. URL https://www.sciencedirect.com/science/article/pii/S0952197625003392.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 5234–5243, Red Hook, NY, USA, 2018. Curran Associates Inc.

C. Sakar and Yomi Kastro. Online Shoppers Purchasing Intention Dataset. UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C5F88Q.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. Tabdiff: a mixed-type diffusion model for tabular data generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Mihaela C Stoian and Eleonora Giunchiglia. Beyond the convexity assumption: Realistic tabular data generation under quantifier-free real linear constraints. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mihaela C Stoian, Salijona Dyrmishi, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How realistic is your synthetic data? constraining deep generative models for tabular data. In *The Twelfth International Conference on Learning Representations*, 2024.

Mihaela Cătălina Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. A survey on tabular data generation: Utility, alignment, fidelity, privacy, and beyond, 2025. URL https://arxiv.org/abs/2503.05954.

Hrithik Suresh, Sahil Sidheekh, Sriraam Natarajan, Narayanan C Krishnan, et al. Tractable sharpness-aware learning of probabilistic circuits. *arXiv preprint arXiv:2508.05537*, 2025.

The april Lab. cirkit, October 2024. URL https://github.com/april-tools/cirkit.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Jakub M Tomczak. *Deep Generative Modeling*. Springer Nature, 2024.

Davide Tugnoli, Andrea De Lorenzo, Marco Virgolin, and Giovanni Cinà. Improving tabpfn's synthetic data generation by integrating causal structure, 2026. URL https://arxiv.org/abs/2603.10254.

Isabel Valera and Zoubin Ghahramani. Automatic discovery of the statistical types of variables in a dataset. In *International Conference on Machine Learning*, pages 3521–3529. PMLR, 2017.

Boris Van Breugel and Mihaela Van Der Schaar. Position: Why tabular foundation models should be a research priority. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 48976–48993. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/van-breugel24a.html.

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 343–358. Springer, 2015.

Antonio Vergari, Nicola Di Mauro, and Guy Van den Broeck. Tractable probabilistic models: Representations,

algorithms, learning, and applications, 2019. In *Tutorial at the 35th Conference on Uncertainty in Artificial Intelligence (UAI 2019)*, 2019a.

Antonio Vergari, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera. Automatic bayesian density analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5207–5215, 2019b.

Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. In *Tutorial at the The 34th AAAI Conference on Artificial Intelligence*, 2020.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 13189–13201. Curran Associates, Inc., 2021.

Stanley. Wasserman and Katherine. Faust. *Social network analysis : methods and applications*. Structural analysis in the social sciences ; 8. Cambridge University Press, Cambridge, 1994. ISBN 0521382696.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 5357–5375. PMLR, 25–27 Apr 2023.

Thomas Wedenig and Robert Peharz. Effective diffusion-free score matching for exact conditional sampling. In *Eighth Workshop on Tractable Probabilistic Modeling*, 2025.

Christopher K. I. Williams. On suspicious coincidences and pointwise mutual information. *Neural Computation*, 34(10):2037–2046, 09 2022. ISSN 0899-7667. doi: 10.1162/neco_a_01533. URL https://doi.org/10.1162/neco_a_01533.

I. H. Witten, Eibe. Frank, and Mark A. Hall. *Data mining : practical machine learning tools and techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier/Morgan Kaufmann, Amsterdam, 3rd ed. edition, 2011. ISBN 9786612953880.

Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, page 7335–7345, 2019.

Scott Cheng-Hsin Yang, Baxter Eaves, Michael Schmidt, Ken Swanson, and Patrick Shafto. Structured evaluation of synthetic tabular data, 2024. URL https://arxiv.org/abs/2403.10424.

Zexi Yao, Nataša Krčo, Georgi Ganev, and Yves-Alexandre de Montjoye. The DCR delusion: Measuring the privacy risk of synthetic data. In *Computer Security – ESORICS 2025: 30th European Symposium on Research in Computer Security, Toulouse, France, September 22–24, 2025, Proceedings, Part I*, pages 469–487. Springer-Verlag, 2025. ISBN 978-3-032-07883-4. doi: 10.1007/978-3-032-07884-1_24. URL https://doi.org/10.1007/978-3-032-07884-1_24.

I-Cheng Yeh. Default of Credit Card Clients. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C55S3H.

EL Hacen Zein and Tanguy Urvoy. Tabular data generation: Can we fool XGBoost ? In *NeurIPS 2022 First Table Representation Workshop*, 2022. URL https://openreview.net/forum?id=tTQzJ6TJGVi.

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

Honghua Zhang, Meihua Dang, Benjie Wang, Stefano Ermon, Nanyun Peng, and Guy Van den Broeck. Scaling probabilistic circuits via monarch matrices. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=uzVShD7wHO.

Han Zhao, Pascal Poupart, and Geoffrey J Gordon. A unified approach for learning the parameters of sum-product networks. *Advances in neural information processing systems*, 29, 2016.

A Sobering Look at Tabular Data Generation via Probabilistic Circuits (Supplementary Material)

Davide Scassola[2, 3, *] Dylan Ponsford[1, *] Adrián Javaloy[1] Sebastiano Saccani[3] Luca Bortolussi[2]
Henry Gouk[1] Antonio Vergari[1]

[1]School of Informatics, University of Edinburgh, Edinburgh, UK
[2]AILAB, University of Trieste, Trieste, Italy
[3]Aindo SpA, AREA Science Park, Trieste, Italy,

A PROOFS

In this section, we prove Thm. 2.1, firstly by proving the following lemma.

Lemma A.1. *Let $\mathcal{D}_p$, $\mathcal{D}_q$ with $|\mathcal{D}_p| = |\mathcal{D}_q| = N$ be two sets of datapoints drawn i.i.d. from distributions p and q respectively. If*

$$\frac{1}{N}\sum_{\boldsymbol{x}\in\mathcal{D}_p}\boldsymbol{x} = \frac{1}{N}\sum_{\boldsymbol{x}\in\mathcal{D}_q}\boldsymbol{x}, \tag{6}$$

i.e. the means of $\mathcal{D}_p$ and $\mathcal{D}_q$ are equal, then the LR classifier learned by maximum likelihood estimation to distinguish $\mathcal{D}_p$ and $\mathcal{D}_q$ is the random classifier.

Proof. Let $y = 1$ denote the case that a datapoint $\boldsymbol{x}$ comes from p, and $y = 0$ denote the case that it comes from q. We form a dataset $\mathcal{D} = \{(\boldsymbol{x}, 0) \,|\, \boldsymbol{x} \in \mathcal{D}_q\} \cup \{(\boldsymbol{x}, 1) \,|\, \boldsymbol{x} \in \mathcal{D}_p\}$, on which to train the logistic regressor classifier $c_{\boldsymbol{w}}(y = 1|\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$, where σ denotes the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$.

When training with maximum likelihood estimation, we want the classifier to maximise the data log-likelihood. In this case, the data log-likelihood is given by

$$\begin{aligned}
\mathcal{L}(\mathcal{D}, \boldsymbol{w}) &= \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \log c_{\boldsymbol{w}}(y \,|\, \boldsymbol{x}), && (7)\\
&= \sum_{\boldsymbol{x}\in\mathcal{D}_p} \log c_{\boldsymbol{w}}(y = 1 \,|\, \boldsymbol{x}) + \sum_{\boldsymbol{x}\in\mathcal{D}_q} \log c_{\boldsymbol{w}}(y = 0 \,|\, \boldsymbol{x}), && (8)\\
&= \sum_{\boldsymbol{x}\in\mathcal{D}_p} \log \sigma(\boldsymbol{w}^T\boldsymbol{x}) + \sum_{\boldsymbol{x}\in\mathcal{D}_q} \log(1 - \sigma(\boldsymbol{w}^T\boldsymbol{x})). && (9)
\end{aligned}$$

Taking the gradient with respect to $\boldsymbol{w}$, we therefore have by linearity that

$$\begin{aligned}
\nabla_{\boldsymbol{w}}\mathcal{L}(\mathcal{D}, \boldsymbol{w}) &= \sum_{\boldsymbol{x}\in\mathcal{D}_p} \nabla_{\boldsymbol{w}} \left(\log \sigma(\boldsymbol{w}^T\boldsymbol{x})\right) + \sum_{\boldsymbol{x}\in\mathcal{D}_q} \nabla_{\boldsymbol{w}} \left(\log(1 - \sigma(\boldsymbol{w}^T\boldsymbol{x}))\right), && (10)\\
&= \sum_{\boldsymbol{x}\in\mathcal{D}_p} \left(1 - \sigma(\boldsymbol{w}^T\boldsymbol{x})\right)\boldsymbol{x} - \sum_{\boldsymbol{x}\in\mathcal{D}_q} \sigma(\boldsymbol{w}^T\boldsymbol{x})\boldsymbol{x}, && (11)
\end{aligned}$$

where Eq. (11) follows from Eq. (10) by the chain rule and due to the fact that $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.

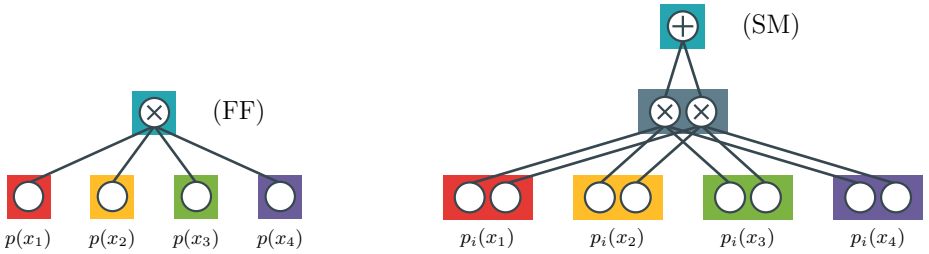


Figure 8: **The simple FF and SM models are special cases of probabilistic circuits.** The FF simply puts a product unit over independent input distributions. The SM mixes together with a sum unit K separate FF models (here, $K = 2$).

Setting the weights to be zero, $\boldsymbol{w} = \mathbf{0}$, we see that

$$\nabla_{\boldsymbol{w}} \mathcal{L}(\mathcal{D}, \boldsymbol{w} = \mathbf{0}) = \sum_{\boldsymbol{x} \in \mathcal{D}_p} \left(1 - \sigma(\mathbf{0}^T \boldsymbol{x})\right) \boldsymbol{x} - \sum_{\boldsymbol{x} \in \mathcal{D}_q} \sigma(\mathbf{0}^T \boldsymbol{x}) \boldsymbol{x}, \tag{12}$$

$$= \sum_{\boldsymbol{x} \in \mathcal{D}_p} \left(1 - \frac{1}{2}\right) \boldsymbol{x} - \sum_{\boldsymbol{x} \in \mathcal{D}_q} \frac{1}{2} \boldsymbol{x}, \tag{13}$$

$$= \frac{1}{2} \left(\sum_{\boldsymbol{x} \in \mathcal{D}_p} \boldsymbol{x} - \sum_{\boldsymbol{x} \in \mathcal{D}_q} \boldsymbol{x} \right), \tag{14}$$

$$= \mathbf{0}, \tag{15}$$

where the last step holds by Eq. (6). Therefore, $\boldsymbol{w} = \mathbf{0}$ is a stationary point of the log-likelihood. Now, since the log-likelihood function of logistic regression is concave, $\boldsymbol{w} = \mathbf{0}$ is hence the unique global maximum. The logistic regressor classifier with weights $\mathbf{0}$ assigns probability $1/2$ to all points, i.e. is the random classifier. The classifier learned by MLE is hence the random classifier. □

We use this lemma to prove Thm. 2.1, which we restate here for convenience.

Theorem A.2. *Let $\mathcal{D}_R$ with $|\mathcal{D}_R| = n$ be a real dataset, and let p denote a FF model trained by MLE on this dataset. Then*

$$\lim_{n \to \infty} \mathbb{E}_{\mathcal{D}_S}[\text{C2ST}(LR, \mathcal{D}_R, \mathcal{D}_S)] = 1,$$

where $\mathcal{D}_S$ is an i.i.d. sample of n items drawn from p.

Proof. We learn FF models with MLE by matching the empirical sufficient statistics. Therefore, in the limit of $n \to \infty$, we have that $\frac{1}{n} \sum_{\mathcal{D}_S} \boldsymbol{x}$ (i.e. the sample mean) converges to the original mean $\frac{1}{n} \sum_{\mathcal{D}_R} \boldsymbol{x}$ for any dataset $\mathcal{D}_S$ of i.i.d. samples from p. By applying Lemma A.1, in the limit as $n \to \infty$ we learn the random classifier with weights $\boldsymbol{w} = \mathbf{0}$. Now, the expected AUROC of the random classifier is 0.5, and therefore the expected C2ST score in the limit is 1. □

B PROBABILISTIC CIRCUITS

B.1 CHOW-LIU ALGORITHM AND REGION GRAPH

The region graph (RG) tells us how to build our PC; in the interpretation as a deep, hierarchical mixture model, the RG instructs us at what depth to mix together different components. If the RG is built carefully, the PC we construct will be smooth and decomposable by design.

Each node in the RG is either a region $\mathcal{R}$, denoting a subset $\mathcal{R} \subseteq \boldsymbol{x}$, or a partition, which describes how a region is partitioned into other regions, $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, with $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$. The root of the RG must necessarily be the 'full' region $\mathcal{R} = \boldsymbol{x}$. So to construct an RG, we need a way of choosing how to partition regions at each level.

When using the Chow-Liu algorithm [Chow and Liu, 1968] to build a region graph, we first build the Chow-Liu tree based on the training data. The Chow-Liu tree is built by computing the pairwise mutual information between all pairs of features.

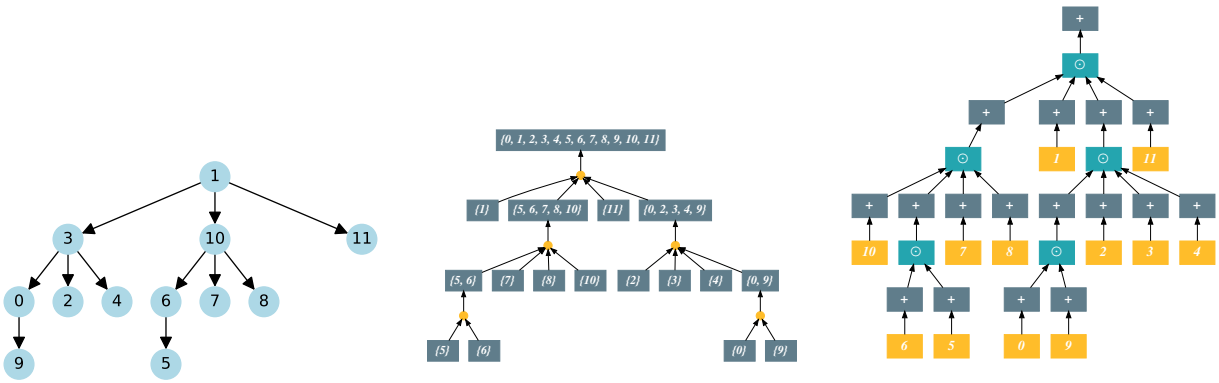


Figure 9: **Circuits can be built directly from a learned Chow-Liu tree**, as we see in the pipeline shown here for the Magic dataset. We first learn a Chow-Liu tree from the training data and root it at its barycentre *(left)*, then compile this into a region graph *(centre)*, then from this we build a circuit *(right)*, where here we choose CP sum-product layers.

This yields a fully connected weighted graph between all features. After that, it builds the maximum spanning tree by using Kruskal's algorithm [Kruskal, 1956]. This yields the Chow-Liu tree (CLT).

Once we have the CLT, this can be compiled directly into a region graph. This process works as follows. We obtain the Jordan centre [Gadat et al., 2018, Floyd, 1962, Wasserman and Faust, 1994] of the CLT, and select this as the root. Then, we build the region graph by progressively merging scopes, starting from the leaf nodes and moving to the root. An example of this is shown above in Fig. 9. For more details, see Dang et al. 2021, Liu and Van den Broeck 2021, Loconte et al. 2025a.

B.2 SAMPLING (CONDITIONALLY) FROM SMOOTH AND DECOMPOSABLE PCS

Sampling from smooth and decomposable PCs can be done via ancestral sampling, by using the latent variable interpretation of each sum unit. This corresponds to a backward traversal of its computational graph. Starting at the output unit, at each sum unit we sample one input branch proportionally to its weight, and then continue traversing the graph towards the inputs. At each product unit, we traverse the graph along all input branches. Once we reach an input unit, we sample from this as normal. Assuming smoothness and decomposability, we are guaranteed to end up in a set of input units whose scope is $\boldsymbol{x}$ and where only one input unit (or rather, input layer) is selected per variable.

We can additionally perform conditional sampling if the input units support tractable conditional sampling. Overall, conditional sampling can be performed by first performing a forward pass with the observed evidence, using this information to modify the sum weights, and then performing the same ancestral sampling as before using these new weights. During this forward pass, the input units either propagate their encoded function evaluated at the observed value (if their scope variable is observed) or 1 (since we are marginalising out unobserved variables). Then, we use these modified weights and perform sampling as before. When we reach an input unit whose scope is observed, instead of sampling we take the observed value of its scope variable (assuming univariate scope).

This procedure is detailed in Alg. 1, which is adapted from Grivas et al. 2026. Here, we additionally assume for simplicity that input units have univariate scope, but the algorithm can be extended to consider multivariate inputs.

This algorithm relies on the following observations: for a smooth and decomposable PC, the conditional of a sum unit is a sum unit over conditional inputs, and the conditional of a product unit is a product unit over conditional inputs. Further details can be found in `https://github.com/smatmo/ESSAI24-PCs/blob/master/lecture01/lecture01.pdf` (slides 37–42).

B.3 ADDITIONAL DETAILS ON TENSORISING AND TRAINING PCS

Following the choice of region graph, degree of overparameterisation, and choice of sum-product layer, the final step in circuit construction pipeline is to *fold* the PC. This means stacking layers with the same functional form so that they can be evaluated in parallel, yielding significant computational benefits [Loconte et al., 2025a]. Note that folding does not change

Algorithm 1 CONDITIONALSAMPLE(c)

Input: A smooth, decomposable and normalised PC c encoding a joint distribution over $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ and observed values $\boldsymbol{X}_o = \boldsymbol{x}_o$ for $\boldsymbol{X}_o \subseteq \boldsymbol{X}$.
Output: a sample $\boldsymbol{x}_m \sim c(\boldsymbol{X}_m | \boldsymbol{X}_o)$ for $\boldsymbol{X}_m = \boldsymbol{X} \setminus \boldsymbol{X}_o$.

for $c_n \in \mathsf{FeedforwardOrder(c)}$ **do** ▷ first compute $c(\boldsymbol{x}_o)$ and store intermediate results
  if c_n is an input unit over variable $X_i \notin \boldsymbol{X}_o$ **then**
    $r_n \leftarrow 1$
  else if c_n is an input unit over variable $X_i \in \boldsymbol{X}_o$ **then**
    $r_n \leftarrow c_n(x_{\phi(n)})$ ▷ compute probability of observation; $\phi(n)$ outputs c_n's scope variable
  else if c_n is a sum unit **then**
    $r_n \leftarrow \sum_{j \in \mathsf{in}(n)} \omega_j r_j$ ▷ $\mathsf{in}(n)$ denotes the input units to c_n
  else if c_n is a product unit **then**
    $r_n \leftarrow \prod_{j \in \mathsf{in}(n)} r_j$
  end if
end for
$\boldsymbol{x} \leftarrow \mathsf{zeroes}(n)$ ▷ init empty sample
$c_n \leftarrow \mathsf{output}(c)$
$\mathcal{N} \leftarrow \mathsf{queue}(\{c_n\})$ ▷ traverse the computational graph from outputs to inputs
while $\mathcal{N}$ not empty **do**
  $c_n \leftarrow \mathsf{pop}(\mathcal{N})$
  if $c_n = \sum_{j=1}^{K} \omega_j c_j$ **then** ▷ c_n is a sum unit with K inputs
    for $j = 1 \ldots K$ **do**
      $\widetilde{\omega}_j \leftarrow \omega_j r_j / \sum_{i=1}^{K} \omega_i r_i$ ▷ modify sum weights by conditioning information
    end for
    $k \leftarrow \mathsf{sampleCategorical}(\widetilde{\omega}_1, \ldots, \widetilde{\omega}_K)$ ▷ sample from a categorical with K states
    $\mathcal{N} \leftarrow \mathsf{push}(\mathcal{N}, c_k)$
  else if $c_n = \prod_{j=1}^{d} c_j$ **then** ▷ c_n is a product unit with d inputs
    for $k = 1 \ldots d$ **do**
      $\mathcal{N} \leftarrow \mathsf{push}(\mathcal{N}, c_k)$ ▷ visit all inputs of c_n
    end for
  else if c_n is an input unit over variable $X_i \notin \boldsymbol{X}_o$ with parameters ϕ_i **then**
    $x_i \leftarrow \mathsf{sampleUnit}(\phi_i)$ ▷ sample from the input distribution if unobserved
  else if c_n is an input unit over variable $X_i \in \boldsymbol{X}_o$ **then**
    $x_i \leftarrow x_{o,i}$ ▷ otherwise take observed value
  end if
end while
return $\boldsymbol{x}$

expressivity — it just enables more efficient computation. For TABPC, we use the default folding algorithm provided in the `cirkit` package [The april Lab, 2024].[1] All later experiments are also implemented using this package.

The training objective for PCs is to directly maximise the log-likelihood (LL) of the training data under the model, which can be tractably evaluated in a single forward pass (if the PC is smooth and decomposable). The LL can be optimised by stochastic gradient ascent (SGD), or by the more specific expectation maximisation (EM), which has been derived for PCs [Peharz et al., 2017, 2020a]. In our experiments, we train using the RAdam optimiser [Liu et al., 2019], since EM is not currently implemented in `cirkit`.

C MORE RELATED WORKS

Before the recent spate of works on TDG mentioned in the main text, some earlier works focused in greater detail on using VAEs. These works were not necessarily framed under the modern banner of TDG, but typically tended to focus on missing data imputation and various tasks associated with this. Such works include Ivanov et al. 2019, Ma et al. 2020, Collier et al.

[1] https://github.com/april-tools/cirkit

2020, Ma et al. 2018, Mattei and Frellsen 2019.

Another related line of research tackles imposing semantic constraints over the generated data [Stoian et al., 2024, Stoian and Giunchiglia, 2025] and evaluating their violation by DGMs. While TABPC is not designed to satisfy constraints, works on using PCs and other tractable models for neuro-symbolic constraint satisfaction could be investigated in the future [Ahmed et al., 2022, Kurscheidt et al., 2025].

Some prior works have also focused in some detail on investigating suitable measures of association between random variables, in particular for the 2×2 contingency table [Williams, 2022, Edwards, 1963, Hasenclever and Scholz, 2016]. Since we consider a more general case, we focus in the main text on comparing TREND and WNMIS, but note that adapting such discussions to evaluating synthetic tabular data could be an interesting avenue for future work.

D EXPERIMENTAL SETUP

We release the code, and will release all final model checkpoints and generated datasets for reproducibility. The code is available at `https://github.com/april-tools/tabpc` and will be updated with the link to checkpoints and datasets.

D.1 DATASET DETAILS

Table 1: Dataset statistics. # Num = number of numerical columns, # Cat = number of categorical columns, # Max Cat = max categories in any categorical column. # Train and # Test refer to the number of samples in the training and test splits respectively.

Dataset	# Rows	# Num	# Cat	# Max Cat	# Train	# Test	Task
Adult [Becker and Kohavi, 1996]	48,842	6	9	42	32,561	16,281	Classification
Beijing [Chen, 2015]	43,824	7	5	31	39,441	4,383	Regression
Default [Yeh, 2009]	30,000	14	11	11	27,000	3,000	Classification
Diabetes [Clore et al., 2014]	101,766	9	27	716	81,412	20,354	Classification
Magic [Bock, 2004]	19,019	10	1	2	17,117	1,902	Classification
News [Fernandes et al., 2015]	39,644	46	2	7	35,679	3,965	Regression
Shoppers [Sakar and Kastro, 2018]	12,330	10	8	20	11,097	1,233	Classification

D.1.1 Dataset Splits

We follow the same dataset splitting as performed by Shi et al. 2025, who in turn follow the protocol of Zhang et al. 2024. Namely, each dataset above is split into train and test sets, with 90% of the data forming the training set, and 10% forming the test set. From the training set, we then form a validation set comprising 10% of this data. Random splits are performed based on the random seed 1234.

D.2 DATASET PREPROCESSING

For comparability, we follow the same pre-processing protocol as in Shi et al. 2025 for removing missing data. Specifically, we replace missing numerical values with the column mean, and treat missing categorical values as a new category. However, we note that as tractable models, PCs would be able to seamlessly handle missing data during training by marginalising out the missing variable(s).

The following pre-processing steps are taken following this missing-value protocol.

For FF, we include two data pre-processing steps.

1. **Inflated value handling:** Inflated values are specific values which are significantly oversampled in the original data; for example, these could be the maximum or minimum possible values. A common example of an inflated value is

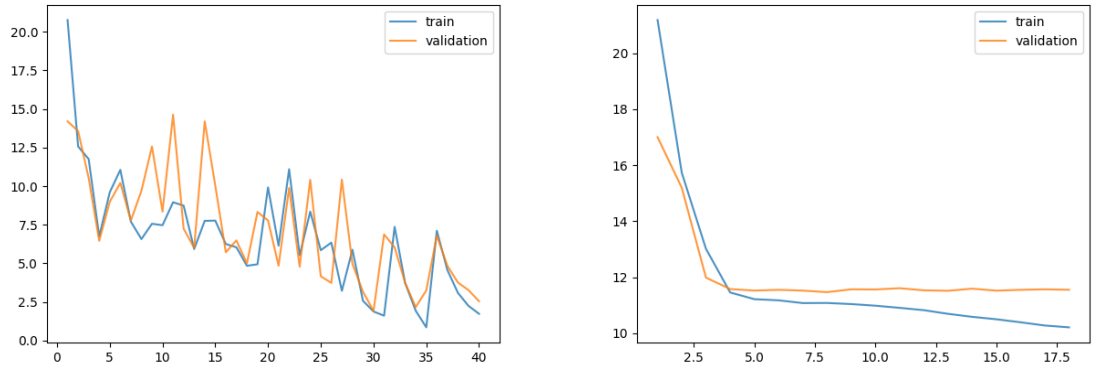


Figure 10: **Dataset pre-processing improves the stability of training.** The plots display training (blue) and validation (orange) negative log-likelihoods on the Adult dataset. Curves are shown for the base model with no pre-processing *(left)*, and for the model with the full pre-processing steps of inflated value handling and quantile normalisation *(right)*. The numerical values for the log-likelihood change when applying the transformation since this changes the density scale.

zero (e.g. the minimum value on some measuring device [Rožanec et al., 2025]). In the case of FF, we simply discard inflated values during fitting.

2. **Quantile normalisation:** this transforms samples from any distribution into samples from a standard Gaussian, and can later be inverted to obtain samples from the data distribution. More details below in App. D.2.1.

For TABPC, we apply the same quantile normalization pre-processing. However, we modify the inflated value handling step.

1. **Inflated value handling:** For TABPC, we handle features with inflated values by creating a new category indicating that an inflated value is present, and then treating the value itself as though it were missing (i.e. by marginalising out that feature). Recall that, with a smooth and decomposable PC, we can efficiently and exactly marginalise out this feature.
2. **Quantile normalisation:** as before.

D.2.1 Quantile Normalisation

We use the `QuantileTransformer` from `scikit-learn` [Pedregosa et al., 2011].[2] This works by estimating the cumulative distribution function (CDF) of the data, and then uses this to map the data to samples from a $\mathrm{Uniform}([0, 1])$. Then, the quantile function of the normal distribution is used to transform the samples to those from a standard Gaussian. This transformation is applied independently to each feature. We go from samples in the normal space to samples in the data space by inverting the transformation.

D.2.2 Preprocessing Ablation Studies

Here we study the effect of these different pre-processing steps on model performance. The results of these studies can be found in Tab. 2. We note here the importance of quantile normalisation in model performance, supporting its use by other methods [Zhang et al., 2024, Shi et al., 2025] as an important pre-processing step. We also provide training curves with and without the full pre-processing, which show that the stability of training is greatly improved with the pre-processing, in Fig. 10.

[2]For further details, see `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html`.

Table 2: Ablation results for pre-processing components. Base denotes no pre-processing, IV denotes inflated value handling, QN denotes quantile normalisation, and IV + QN denotes having both of these components. We observe in particular that quantile normalisation is an important component of model performance over the base model—all datasets except Diabetes show a benefit when enabling quantile normalisation.

Dataset	**Pre-processing**	**C2ST (XGB)**
Adult	Base	0.0080
Adult	IV	0.6239
Adult	QN	0.7544
Adult	IV + QN	**0.8910**
Beijing	Base	0.0068
Beijing	IV	0.0072
Beijing	QN	**0.6088**
Beijing	IV + QN	0.5744
Default	Base	0.0103
Default	IV	0.0200
Default	QN	0.2719
Default	IV + QN	**0.2799**
Diabetes	Base	**0.7968**
Diabetes	IV	0.7915
Diabetes	QN	0.7884
Diabetes	IV + QN	0.7889
Magic	Base	0.7721
Magic	IV	0.7743
Magic	QN	0.8585
Magic	IV + QN	**0.8627**
News	Base	0.0000
News	IV	0.0000
News	QN	**0.1588**
News	IV + QN	0.1132
Shoppers	Base	0.0022
Shoppers	IV	0.5822
Shoppers	QN	0.8194
Shoppers	IV + QN	**0.8866**

D.3 METRIC DETAILS

D.3.1 SHAPE

The SHAPE metric, implemented in `SDMetrics` [Dat, 2025], captures how well synthetic data models the univariate marginal distributions of the real data. The way it is computed for each feature depends on that feature's type.

For numerical features, the *Kolmogorov-Smirnov statistic* is used. This compares the two empirical cumulative distribution functions (CDFs) of the univariate marginals in both the real and synthetic data. Denoting these by $\widehat{F}_x$ and $\widehat{G}_x$ for feature x respectively, the Kolmogorov-Smirnov statistic is defined for feature x by [Massey Jr., 1951]

$$D_{KS}\left(\widehat{F}_x, \widehat{G}_x\right) = \max_y \left|\widehat{F}_x(y) - \widehat{G}_x(y)\right| \in [0,1], \tag{16}$$

i.e. the maximum of the set of distances between the CDFs. This difference is subtracted from 1 to give a similarity score to the feature, so for numerical feature x, its corresponding score is given by

$$s_{\text{numerical}}(x) = 1 - D_{KS}\left(\widehat{F}_x, \widehat{G}_x\right). \tag{17}$$

This score is also in $[0,1]$, where 1 is the maximum.

For categorical features, the *total variation distance* (TVD) is used. This computes the difference between the empirical probabilities as follows:

$$\text{TVD}(R_x, S_x) = \frac{1}{2}\sum_{\alpha \in x} |R_\alpha - S_\alpha| \in [0,1], \tag{18}$$

where the sum over α ranges over the possible categories of x, and R_α, S_α denote the empirical probabilities of $x = \alpha$ in the real and synthetic datasets respectively. This distance is again subtracted from 1 to yield a similarity score:

$$s_{\text{categorical}}(x) = 1 - \text{TVD}(R_x, S_x) \tag{19}$$

Again, this score is also in $[0,1]$, where 1 is the maximum.

The overall SHAPE score is then given by the average scores of all columns (features) in the dataset.

D.3.2 α-PRECISION and β-RECALL

α-PRECISION and β-RECALL are proposed by Alaa et al. 2022, extending the work by Sajjadi et al. 2018 on assessing the precision and recall of generative models. These are sample-level metrics quantifying the sample's 'fidelity' and 'diversity'. In this case, they use fidelity to refer to sample quality ('realism' of samples), similarly to before, and diversity to refer to how well the samples cover the variability of the real data. Since these are sample-level metrics, they can be averaged over the entire synthetic dataset to provide an overall score.

They work by embedding the real and synthetic data into hyperspheres via a learned one-class neural network, where most samples are concentrated in the centre (so that the supports of the real and generated data are hyperspherical). Then, they use minimum volume sets covering a proportion (α or β) of the (real or synthetic) data, which in the hypersphere embedding space are given by hyperspheres of certain radii. For a given proportion α, its α-support describes the minimum volume subset of its support containing probability mass α.

For α-PRECISION, we want to know whether the synthetic sample falls within the α-support of the real data. This tells us whether a synthetic sample is typical (i.e. whether it could come from the real dataset). For β-RECALL, we want to know whether the real samples fall within the β-support of the synthetic data. This tells us whether the synthetic data covers the full variability of real samples. To compute these, we train classifiers to predict whether a given sample is contained within the α- or β-support.

For a full technical description of these metrics, see Alaa et al. 2022.

One important consideration is that the actual implementation for these metrics can sometimes differ in recent related works to the approach described above; this is described in further detail in App. I.

D.4 BASELINE DETAILS

For comparison, we use the codebase provided by Shi et al. 2025 for TabDiff,[3] and the codebase provided by Zhang et al. 2024 for all other mentioned methods.[4] We also use their provided default hyperparameters.

Other Baselines: We were unable to reproduce previous results for the methods GOGGLE [Liu et al., 2023b] and TabDDPM [Kotelnikov et al., 2023] due to environment and training issues. In particular, TabDDPM has previously been criticised for such issues on some datasets like Diabetes [Shi et al., 2025], but we experienced issues across all datasets. Moreover, we were unable to compare against the newer flow-based method TabbyFlow [Guzmán-Cordero et al., 2025], since at the original time of writing there appeared to be errors in the sampling code released online.[5] Since then, the repo has been updated, but we found further issues in trying to get the codebase to run.[6]

D.4.1 TabPFN Further Details

TabPFN is a transformer-based model pre-trained on a large amount of synthetic data generated by Structural Causal Models [Pearl, 2009]. It shifts the discriminative paradigm to an in-context learning (ICL) approach, and is a 'Prior-data Fitted Network (PFN)' performing approximate Bayesian inference [Müller et al., 2022]. When using TabPFN for discriminative tasks, the dataset is passed to the model, embedded, and then the model outputs predictions directly in a single forward pass (see e.g. Hollmann et al. 2023). In particular, no training or fine-tuning is required.

For TabPFN, we installed the version of `tabpfn-extensions` corresponding to commit `7f84343` on GitHub. We used the default version of TabPFN, which is currently v3 [Grinsztajn et al., 2026]. Additionally, we follow the package's default value of three feature ordering permutations for our data generation experiments, following concurrent work [Tugnoli et al., 2026]. While other approaches to using TabPFN for TDG exist [Ma et al., 2023], we choose to use `tabpfn-extensions` as the 'most official' implementation.

When no features have yet been sampled, TabPFN conditions on random Gaussian noise so as to approximate the first feature's marginal distribution. After that, features are autoregressively generated based on the current feature ordering permutation [Tugnoli et al., 2026].

The documentation of TabPFN specifies that minimal pre-processing should be applied. Hence we apply only an ordinal encoding to categorical features. Moreover, we specify the categorical features to TabPFN as those with fewer than 160 unique values (the maximum category count of TabPFN v3). This is a reasonable heuristic given the number of samples our datasets have (> 1000, see Tab. 1).

Diabetes has a feature which exceeds this maximum number of categories, and so we were not able to generate data for this dataset. Moreover, Magic contained one failed run, and News contained three failed runs, so the standard deviations for these datasets are reported over four runs and two runs respectively (by failed, we mean a run which failed more than 3 times, which we set as our 'patience'). The error in each case related to generating a NaN for one of the numerical features.

It is possible that this instability during generation is caused by some underlying bug in the implementation, as it appears that other users have previously run into a similar issue (`https://github.com/PriorLabs/tabpfn-extensions/issues/120`). Since this is package of community extensions, the documentation specifies that the functionality is less rigorously tested than the main TabPFN library, so this remains a plausible explanation.

D.5 TABPC FURTHER DETAILS

Hyperparameter configurations for TabPC on each dataset can be found in Tab. 3.

When training TabPC (alongside the hyperparameters mentioned above), we use a learning rate scheduler which decreases the learning rate by a factor of 0.85 if the validation log-likelihood has not decreased in the last epoch. Recall that we use the RAdam optimiser [Liu et al., 2019] for training. We also use the validation set (recall that this is a 10% split of the training set) to perform early stopping, with a patience of 10 epochs. Columns with fewer than 50 unique values are treated

[3] `https://github.com/MinkaiXu/TabDiff`
[4] `https://github.com/amazon-science/tabsyn`
[5] E.g. `https://github.com/andresguzco/ef-vfm/issues/1`
[6] See `https://github.com/andresguzco/ef-vfm/issues/4`.

Table 3: TABPC hyperparameters (number of sum and input units, batch size, and learning rate) and number of shallow mixture components for each dataset. Values were found by evaluating the BPD of trained instances of TABPC on the respective validation set.

Dataset	TABPC			Shallow Mixture
	# Units	Batch Size	Learning Rate	# Components (K)
Adult	4096	512	0.25	20 000
Beijing	4096	512	0.25	50 000
Default	512	512	0.10	20 000
Diabetes	2048	512	0.25	20 000
Magic	2048	256	0.10	10 000
News	1024	512	0.10	10 000
Shoppers	2048	512	0.10	20 000

as categorical, and otherwise treated as continuous. Quantised numerical columns are dequantised by adding uniform noise in the range $[-q/2, q/2]$, where q is the quantisation step. After sampling, these are re-quantised to their original grid.

D.6 COMPUTATIONAL RESOURCES

The following hardware was used to train the models and evaluate the results:

- **GPU:** NVIDIA RTX A6000 (49 GiB)
- **Processor:** AMD EPYC 7452 32-Core Processor
- **Memory:** 512 GiB

E EXPERIMENTAL RESULTS

E.1 TABLES

The uncertainties in each table represent the standard deviation of measurements over 5 model seeds. For TABPFN, these are over 5 random generations with the exception of Magic and News, where TABPFN experienced crashed multiple times during these runs (see App. D.4.1). Hence, these standard deviations are over four and two datasets respectively.

In Tab. 11, Beijing and News are associated with regression tasks, with trained models evaluated by root mean squared error (RMSE). The other datasets are associated with classification tasks, evaluated by classifier area under curve (AUC).

Where ranks are assigned, missing entries are given the worst possible rank. The missing entries for GREAT correspond to datasets for which it either failed to train (Diabetes) or generate (News). On some occasions, GREAT was able to generate datasets, but would generate categories which did not appear in the training data, thereby throwing an error when evaluating metrics. STASY technically finished training on Diabetes, but obtains quite poor results.

Entries containing < 0.0001 mean that the five model seeds each achieved a score less than this precision threshold.

E.2 CDDS

In the generated CDDs, the value in the scale represents the average rank of the method (based on the means of the observations for each dataset). Methods are then connected by a solid line if they cannot be statistically distinguished from one another (i.e. are in the same *clique*).

These CDDs are produced using the `aeon` package [Middlehurst et al., 2024]. To compute the cliques, `aeon` uses a one-sided Wilcoxon sign rank test with the Holm correction.[7]

[7]Further details can be found at https://www.aeon-toolkit.org/en/stable/api_reference/auto_generated/aeon.visualisation.plot_critical_difference.html.

Table 4: **TabPC reports strong results on the Shape metric, exceeding the performance of SotA diffusion-based models in six out of the seven datasets.** We note that the FF model in fact obtains higher scores even than TabPC in all except one dataset, likely because matching the marginals is essentially how it is trained.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.8087±0.0191 (8)	0.7999±0.0221 (10)	0.8308±0.0111 (9)	0.8957±0.0042 (6)	0.9443±0.0065 (8)	0.8622±0.0036 (8)	0.7406±0.0174 (10)	8.43
TVAE	0.7619±0.0135 (10)	0.7216±0.0196 (11)	0.9127±0.0023 (8)	0.7618±0.0471 (8)	0.9540±0.0067 (7)	0.8169±0.0083 (9)	0.7508±0.0187 (9)	8.86
GReaT	0.4250±0.0002 (11)	0.9339±0.0010 (7)	0.8047±0.0015 (10)	* (10)	0.8498±0.0009 (11)	* (11)	0.8578±0.0006 (8)	9.71
STaSy	0.8915±0.0172 (7)	0.8916±0.0204 (8)	0.9285±0.0172 (7)	0.3712±0.0000 (9)	0.8817±0.0200 (10)	0.9016±0.0233 (7)	0.8651±0.0291 (7)	7.86
CoDi	0.7662±0.0245 (9)	0.8115±0.0404 (9)	0.7790±0.0300 (11)	0.7868±0.0000 (7)	0.9050±0.0088 (9)	0.7173±0.0146 (10)	0.6671±0.0138 (11)	9.43
TabSyn	0.9921±0.0010 (5)	0.9739±0.0126 (6)	0.9868±0.0043 (5)	0.9822±0.0008 (5)	0.9902±0.0019 (3)	0.9766±0.0122 (4)	0.9852±0.0018 (3)	4.43
TabDiff	0.9932±0.0008 (3)	0.9895±0.0004 (3)	0.9884±0.0016 (4)	0.9855±0.0070 (4)	0.9920±0.0008 (2)	0.9725±0.0047 (5)	0.9851±0.0028 (4)	3.57
TabPFN	0.9734±0.0006 (6)	0.9869±0.0004 (4)	0.9369±0.0005 (6)	* (10)	0.9819±0.0014 (6)	0.9224±0.0001 (6)	0.8846±0.0009 (6)	6.29
FF	0.9961±0.0006 (1)	0.9949±0.0003 (1)	0.9954±0.0003 (1)	0.9966±0.0001 (1)	0.9938±0.0007 (1)	0.9907±0.0001 (2)	0.9934±0.0007 (1)	**1.14**
SM	0.9924±0.0008 (4)	0.9811±0.0046 (5)	0.9907±0.0006 (3)	0.9940±0.0013 (3)	0.9840±0.0012 (5)	0.9880±0.0012 (3)	0.9712±0.0105 (5)	4.00
TabPC	0.9942±0.0007 (2)	0.9943±0.0005 (2)	0.9946±0.0004 (2)	0.9942±0.0004 (2)	0.9902±0.0013 (3)	0.9936±0.0009 (1)	0.9910±0.0014 (2)	2.00

Table 5: As we have described in Sec. 2.2, **Trend is problematic** in that it is easily 'fooled' by the trivial FF model below. For example, on Diabetes, the FF model is able to beat diffusion-based TabSyn.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.7581±0.0205 (9)	0.7430±0.0311 (10)	0.7024±0.0285 (11)	0.8095±0.0170 (6)	0.9445±0.0071 (6)	0.9485±0.0011 (9)	0.7554±0.0189 (11)	8.86
TVAE	0.6664±0.0330 (10)	0.7022±0.0356 (11)	0.8135±0.0287 (8)	0.5936±0.0925 (8)	0.9531±0.0120 (5)	0.9386±0.0064 (10)	0.7928±0.0256 (10)	8.86
GReaT	0.1907±0.0023 (11)	0.9185±0.0272 (7)	0.7765±0.0187 (9)	* (10)	0.9076±0.0094 (9)	* (11)	0.8892±0.0047 (6)	9.00
STaSy	0.8526±0.0204 (7)	0.8804±0.0203 (9)	0.9297±0.0221 (5)	0.1480±0.0000 (9)	0.9399±0.0158 (7)	0.9699±0.0043 (5)	0.8745±0.0259 (7)	7.00
CoDi	0.7714±0.0163 (8)	0.9258±0.0101 (5)	0.8376±0.0119 (7)	0.6859±0.0000 (7)	0.9397±0.0050 (8)	0.9571±0.0008 (6)	0.8059±0.0095 (9)	7.14
TabSyn	0.9795±0.0030 (4)	0.9550±0.0157 (4)	0.9712±0.0094 (3)	0.9603±0.0011 (5)	0.9918±0.0014 (1)	0.9833±0.0044 (2)	0.9771±0.0013 (3)	3.14
TabDiff	0.9846±0.0007 (2)	0.9741±0.0018 (2)	0.9726±0.0068 (2)	0.9686±0.0083 (3)	0.9916±0.0021 (2)	0.9836±0.0026 (1)	0.9809±0.0017 (2)	**2.00**
TabPFN	0.9152±0.0201 (6)	0.9236±0.0363 (6)	0.7699±0.0542 (10)	* (10)	0.8936±0.0139 (10)	0.9494±0.0030 (8)	0.8099±0.0233 (8)	8.29
FF	0.9251±0.0004 (5)	0.9101±0.0010 (8)	0.8875±0.0013 (6)	0.9685±0.0017 (4)	0.8696±0.0005 (11)	0.9548±0.0006 (7)	0.9367±0.0014 (5)	6.57
SM	0.9814±0.0012 (3)	0.9646±0.0062 (3)	0.9771±0.0013 (1)	0.9741±0.0016 (2)	0.9789±0.0010 (4)	0.9758±0.0007 (4)	0.9503±0.0091 (4)	3.00
TabPC	0.9856±0.0012 (1)	0.9781±0.0025 (1)	0.9496±0.0189 (4)	0.9810±0.0019 (1)	0.9830±0.0051 (3)	0.9815±0.0020 (3)	0.9822±0.0023 (1)	**2.00**

Table 6: **TabPC has the highest average rank of all methods on wNMIS**, exceeding the mean performance of the diffusion-based SotA in four out of the seven datasets.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.8423±0.0284 (9)	0.8734±0.0399 (9)	0.8274±0.0110 (10)	0.9455±0.0028 (4)	0.9198±0.0051 (10)	0.8679±0.0020 (8)	0.8921±0.0306 (10)	8.57
TVAE	0.9127±0.0070 (8)	0.8734±0.0314 (9)	0.9599±0.0099 (5)	0.8704±0.0448 (9)	0.9719±0.0024 (8)	0.8793±0.0047 (7)	0.9392±0.0110 (7)	7.57
GReaT	0.9561±0.0005 (6)	0.9837±0.0016 (4)	0.8709±0.0016 (9)	* (10)	0.9723±0.0008 (7)	* (11)	0.9577±0.0027 (6)	7.57
STaSy	0.9735±0.0049 (5)	0.9781±0.0067 (7)	0.9504±0.0172 (7)	0.8917±0.0000 (8)	0.9796±0.0120 (5)	0.9392±0.0063 (2)	0.9602±0.0031 (5)	5.57
CoDi	0.7898±0.0073 (10)	0.9812±0.0077 (6)	0.8927±0.0049 (8)	0.9336±0.0000 (6)	0.9741±0.0035 (6)	0.9232±0.0036 (5)	0.9171±0.0086 (9)	7.14
TabSyn	0.9860±0.0026 (4)	0.9833±0.0076 (5)	0.9846±0.0029 (3)	0.9797±0.0006 (2)	0.9971±0.0003 (1)	0.9441±0.0122 (1)	0.9832±0.0042 (3)	2.71
TabDiff	0.9889±0.0009 (2)	0.9927±0.0004 (2)	0.9881±0.0016 (2)	0.9687±0.0309 (3)	0.9953±0.0016 (2)	0.9127±0.0262 (6)	0.9844±0.0011 (2)	2.71
TabPFN	0.9885±0.0007 (3)	0.9971±0.0005 (1)	0.9648±0.0012 (4)	* (10)	0.9941±0.0010 (3)	0.9249±0.0002 (4)	0.9614±0.0009 (4)	4.14
FF	0.7189±0.0002 (11)	0.7918±0.0001 (11)	0.6942±0.0001 (11)	0.9082±0.0002 (7)	0.7684±0.0000 (11)	0.7421±0.0001 (10)	0.8661±0.0002 (11)	10.29
SM	0.9540±0.0071 (7)	0.9696±0.0032 (8)	0.9573±0.0016 (6)	0.9342±0.0088 (5)	0.9432±0.0037 (9)	0.8175±0.0081 (9)	0.9266±0.0048 (8)	7.43
TabPC	0.9933±0.0049 (1)	0.9925±0.0010 (3)	0.9894±0.0012 (1)	0.9919±0.0024 (1)	0.9925±0.0012 (4)	0.9388±0.0048 (3)	0.9927±0.0009 (1)	**2.00**

Table 7: **C2ST (LR) is flawed and should not be considered a strong indicator of model performance.** This is evidenced by the performance of the FF model, which is able to achieve perfect or almost-perfect scores across a range of datasets.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.5881±0.0928 (7)	0.5534±0.1899 (10)	0.4129±0.0875 (10)	0.5293±0.0693 (6)	0.7966±0.0422 (8)	0.7486±0.0428 (7)	0.5255±0.0319 (7)	7.86
TVAE	0.2286±0.0416 (9)	0.3707±0.0498 (11)	0.6662±0.0182 (7)	0.0177±0.0259 (7)	0.8765±0.0261 (7)	0.4355±0.0321 (9)	0.2728±0.0675 (10)	8.57
GReaT	* (11)	0.7645±0.0040 (7)	0.4815±0.0019 (9)	* (9)	0.4628±0.0027 (11)	* (11)	0.4402±0.0047 (8)	9.43
STaSy	0.4537±0.0268 (8)	0.6531±0.0414 (8)	0.6303±0.0628 (8)	* (9)	0.5496±0.0707 (10)	0.4448±0.1707 (8)	0.3860±0.0979 (9)	8.57
CoDi	0.1986±0.0286 (10)	0.6345±0.2901 (9)	0.3287±0.0602 (11)	0.0023±0.0000 (8)	0.7425±0.0238 (9)	0.1133±0.0860 (10)	0.2169±0.0164 (11)	9.71
TabSyn	0.9866±0.0081 (5)	0.9226±0.0303 (5)	0.9463±0.0588 (5)	0.6617±0.0397 (5)	0.9940±0.0051 (3)	0.9381±0.0629 (4)	0.9787±0.0141 (3)	4.29
TabDiff	0.9899±0.0054 (3)	0.9770±0.0027 (4)	0.9699±0.0095 (4)	0.9290±0.0651 (4)	0.9964±0.0035 (1)	0.9057±0.0588 (5)	0.9756±0.0170 (4)	3.57
TabPFN	0.8864±0.0027 (6)	0.9961±0.0029 (3)	0.8820±0.0068 (6)	* (9)	0.9722±0.0040 (5)	0.8380±0.0009 (6)	0.7170±0.0067 (6)	5.86
FF	1.0000±0.0000 (1)	0.9979±0.0029 (2)	0.9949±0.0045 (2)	1.0000±0.0000 (1)	0.9892±0.0072 (4)	0.9795±0.0132 (2)	1.0000±0.0000 (1)	1.86
SM	0.9879±0.0041 (4)	0.9133±0.0119 (6)	0.9826±0.0054 (3)	0.9993±0.0016 (3)	0.9551±0.0056 (6)	0.9765±0.0128 (3)	0.9106±0.0501 (5)	4.29
TabPC	0.9959±0.0083 (2)	0.9989±0.0015 (1)	0.9999±0.0003 (1)	1.0000±0.0000 (1)	0.9953±0.0067 (2)	0.9960±0.0033 (1)	0.9969±0.0056 (2)	**1.43**

Table 8: **TABPC offers competitive performance on C2ST (XGB),** slightly exceeding the average rank across all datasets of TABDIFF. In Diabetes and Shoppers, TABPC also greatly exceeds the performance of the current SotA, but suffers compared to TABDIFF on Beijing and Default. We note that all datasets struggle on the complex dataset News, which contains many numerical features and skewed distributions. We use the default package parameters for our XGBoost classifier.

Method	Adult		Beijing		Default		Diabetes		Magic		News		Shoppers		Avg. Rank
CTGAN	<0.0001	(8)	<0.0001	(10)	0.0002±0.0001	(10)	<0.0001	(6)	0.1034±0.0057	(10)	<0.0001	(6)	<0.0001	(10)	8.57
TVAE	<0.0001	(8)	<0.0001	(10)	0.0020±0.0002	(8)	<0.0001	(6)	0.2981±0.0276	(8)	<0.0001	(6)	<0.0001	(10)	8.00
GReaT	*	(11)	0.6016±0.0028	(3)	0.2425±0.0012	(5)	*	(9)	0.2702±0.0027	(9)	*	(11)	0.2924±0.0034	(4)	7.43
STaSy	0.3425±0.0272	(6)	0.3668±0.0243	(6)	0.3571±0.0370	(3)	*	(9)	0.5221±0.0790	(6)	0.1917±0.0525	(1)	0.2277±0.0806	(6)	5.29
CoDi	<0.0001±0.0001	(8)	0.0002±0.0001	(9)	0.0014±0.0002	(9)	<0.0001	(6)	0.4201±0.0430	(7)	<0.0001	(6)	0.0002±0.0001	(9)	7.71
TabSyn	0.7942±0.0341	(4)	0.5266±0.1397	(5)	0.4024±0.0515	(2)	0.5222±0.0362	(3)	0.7602±0.0310	(4)	0.1208±0.0590	(3)	0.6554±0.0196	(3)	3.43
TabDiff	0.8409±0.0067	(2)	0.7723±0.0138	(1)	0.5914±0.0226	(1)	0.5618±0.2842	(2)	0.7732±0.0278	(3)	0.0296±0.0252	(4)	0.7361±0.0162	(2)	2.14
TabPFN	0.8257±0.0042	(3)	0.2289±0.0048	(7)	0.0798±0.0003	(7)	*	(9)	0.8572±0.0048	(2)	0.0002±0.0001	(5)	0.0021±0.0002	(8)	5.86
FF	0.0160±0.0007	(7)	0.0172±0.0006	(8)	0.0002±0.0001	(10)	0.0503±0.0009	(5)	0.0052±0.0002	(11)	<0.0001	(6)	0.0211±0.0013	(7)	7.71
SM	0.7727±0.0243	(5)	0.7332±0.0148	(2)	0.1053±0.0127	(6)	0.1329±0.0328	(4)	0.6258±0.0228	(5)	<0.0001±0.0001	(6)	0.2901±0.0251	(5)	4.71
TabPC	0.8554±0.0328	(1)	0.5704±0.0262	(4)	0.2617±0.0052	(4)	0.7915±0.0134	(1)	0.8602±0.0142	(1)	0.1286±0.0182	(2)	0.8936±0.0100	(1)	**2.00**

Table 9: **TABPC offers competitive performance on α-PRECISION compared to diffusion-based SotA**, a measure of sample fidelity, though its average rank is slightly lower than top-performing TABDIFF. However, the error bars for certain datasets like Adult and Beijing overlap, and hence ranking by mean does not tell the full story. Note that the following numbers are **not** backwards compatible with recent works like TABSYN and TABDIFF due to the implementation being different; App. I has further details.

Method	Adult		Beijing		Default		Diabetes		Magic		News		Shoppers		Avg. Rank
CTGAN	0.8951±0.0313	(8)	0.5669±0.1178	(9)	0.9313±0.0372	(10)	0.8381±0.0976	(6)	0.8060±0.0238	(10)	0.2107±0.0023	(3)	0.7908±0.0899	(8)	7.71
TVAE	0.5219±0.0182	(11)	0.9795±0.0137	(2)	0.8898±0.0145	(11)	0.7829±0.0952	(7)	0.9580±0.0350	(5)	0.1927±0.0018	(10)	0.9220±0.0139	(4)	7.14
GReaT	0.9494±0.0002	(7)	*	(11)	0.9694±0.0015	(4)	*	(10)	0.9576±0.0033	(6)	*	(11)	0.7115±0.0044	(9)	8.29
STaSy	0.8573±0.0824	(9)	0.9074±0.0325	(8)	0.9489±0.0358	(7)	<0.0001	(9)	0.9353±0.0547	(8)	0.1943±0.0034	(9)	0.8010±0.1559	(7)	8.14
CoDi	0.8538±0.0172	(10)	0.9664±0.0227	(3)	0.9402±0.0083	(8)	0.2804±0.0000	(8)	0.9890±0.0055	(2)	0.2018±0.0053	(5)	0.6690±0.0133	(10)	6.57
TabSyn	0.9948±0.0027	(2)	0.9375±0.0609	(6)	0.9747±0.0033	(3)	0.9799±0.0047	(2)	0.9863±0.0047	(3)	0.1958±0.0022	(7)	0.9886±0.0049	(1)	3.43
TabDiff	0.9955±0.0011	(1)	0.9477±0.0029	(4)	0.9891±0.0027	(1)	0.9680±0.0445	(3)	0.9820±0.0052	(4)	0.2068±0.0077	(4)	0.9649±0.0050	(3)	**2.86**
TabPFN	0.9781±0.0031	(6)	0.9883±0.0032	(1)	0.9682±0.0028	(6)	*	(10)	0.9941±0.0026	(1)	0.1950±0.0002	(8)	0.9168±0.0040	(5)	5.29
FF	0.9927±0.0026	(4)	0.4089±0.0021	(10)	0.9348±0.0008	(9)	0.9333±0.0029	(5)	0.4475±0.0038	(11)	0.3023±0.0018	(1)	0.5133±0.0076	(11)	7.29
SM	0.9931±0.0016	(3)	0.9322±0.0147	(7)	0.9687±0.0064	(5)	0.9378±0.0022	(4)	0.9181±0.0062	(9)	0.2339±0.0059	(2)	0.8142±0.0305	(6)	5.14
TabPC	0.9912±0.0038	(5)	0.9419±0.0060	(5)	0.9757±0.0060	(2)	0.9833±0.0075	(1)	0.9567±0.0054	(7)	0.1959±0.0003	(6)	0.9875±0.0049	(2)	4.00

Table 10: **TABPC also offers competitive performance on β-RECALL compared to SotA methods**, a measure of sample diversity. One interesting observation is that TABPFN performs relatively higher than it does in other metrics here. As with α-PRECISION, error bars can often overlap. Note that the following numbers are **not** backwards compatible with recent works like TABSYN and TABDIFF due to the implementation being different; App. I has further details.

Method	Adult		Beijing		Default		Diabetes		Magic		News		Shoppers		Avg. Rank
CTGAN	0.4497±0.0486	(11)	0.1670±0.0551	(9)	0.4598±0.0277	(11)	0.3055±0.0107	(6)	0.3249±0.0039	(10)	0.2077±0.0001	(5)	0.2895±0.0190	(9)	8.71
TVAE	0.4547±0.0136	(10)	0.2579±0.0225	(8)	0.4696±0.0114	(8)	0.1463±0.0909	(7)	0.4556±0.0096	(7)	0.2076±0.0010	(6)	0.4103±0.0096	(7)	7.57
GReaT	0.5191±0.0008	(1)	*	(11)	0.4675±0.0025	(10)	*	(10)	0.4439±0.0068	(9)	*	(11)	0.4560±0.0026	(4)	8.00
STaSy	0.5020±0.0102	(5)	0.3859±0.0137	(6)	0.4924±0.0048	(5)	<0.0001	(9)	0.4540±0.0269	(8)	0.2083±0.0009	(1)	0.3752±0.0583	(8)	6.00
CoDi	0.5164±0.0037	(2)	0.3696±0.0178	(7)	0.4841±0.0068	(7)	0.0388±0.0000	(8)	0.4821±0.0091	(5)	0.2078±0.0004	(4)	0.2158±0.0332	(11)	6.29
TabSyn	0.5003±0.0041	(7)	0.4125±0.0336	(4)	0.4940±0.0050	(3)	0.4765±0.0033	(2)	0.4873±0.0048	(2)	0.2075±0.0005	(7)	0.4952±0.0059	(3)	4.00
TabDiff	0.5031±0.0026	(4)	0.4373±0.0046	(2)	0.5018±0.0015	(1)	0.4705±0.0632	(3)	0.4873±0.0025	(2)	0.2075±0.0001	(7)	0.4998±0.0090	(2)	**3.00**
TabPFN	0.5079±0.0027	(3)	0.5204±0.0033	(1)	0.4953±0.0034	(2)	*	(10)	0.4965±0.0016	(1)	0.2080±0.0002	(3)	0.4337±0.0098	(5)	3.57
FF	0.4975±0.0030	(9)	0.0961±0.0030	(10)	0.4689±0.0018	(9)	0.3670±0.0015	(5)	0.1318±0.0026	(11)	0.2046±0.0001	(10)	0.2536±0.0064	(10)	9.14
SM	0.5018±0.0042	(6)	0.4214±0.0059	(3)	0.4925±0.0029	(4)	0.4029±0.0123	(4)	0.4616±0.0065	(6)	0.2073±0.0001	(9)	0.4235±0.0138	(6)	5.43
TabPC	0.4983±0.0027	(8)	0.3886±0.0030	(5)	0.4914±0.0045	(6)	0.4804±0.0027	(1)	0.4839±0.0031	(4)	0.2081±0.0001	(2)	0.5049±0.0035	(1)	3.86

Table 11: **TABPC generates synthetic data which is generally competitive with SotA for training new machine learning models** (in terms of ***machine learning efficacy; MLE***). Values for Beijing and News are root mean squared errors (RMSEs), whereas values for other datasets are classifier area under receiver operating characteristic curves (AUROCs). Metric direction of improvement is indicated by the arrow next to the dataset name. Values should be compared to models which used the real training data, which can be found in the first row.

Method	**Adult** (↑)	**Beijing** (↓)	**Default** (↑)	**Diabetes** (↑)	**Magic** (↑)	**News** (↓)	**Shoppers** (↑)
Real Data	0.9271±0.0006	0.4367±0.0116	0.7697±0.0008	0.7040±0.0016	0.9478±0.0024	0.8403±0.0041	0.9254±0.0010
CTGAN	0.8623±0.0078	1.0337±0.1615	0.7268±0.0158	0.5908±0.0094	0.8823±0.0097	0.8603±0.0179	0.8501±0.0168
TVAE	0.8710±0.0152	1.0378±0.0376	0.7467±0.0043	0.5798±0.0221	0.9181±0.0067	0.9831±0.0309	0.9032±0.0095
GReaT	0.8341±0.0051	0.6216±0.0142	0.7584±0.0055	*	0.9114±0.0053	*	0.9067±0.0039
STaSy	0.9058±0.0010	0.6761±0.0417	0.7534±0.0075	*	0.9320±0.0038	0.9222±0.1028	0.9103±0.0057
CoDi	0.8048±0.0353	0.8082±0.0580	0.5030±0.0107	0.4778±0.0000	0.9318±0.0048	2.1437±0.7155	0.8640±0.0214
TabSyn	0.9104±0.0014	0.6260±0.0496	0.7590±0.0058	0.6862±0.0032	0.9370±0.0022	**0.8577**±0.0312	**0.9144**±0.0071
TabDiff	0.9130±0.0013	0.5642±0.0123	**0.7625**±0.0064	0.6806±0.0242	0.9358±0.0050	0.8784±0.0068	**0.9170**±0.0043
TabPFN	0.9121±0.0010	**0.4780**±0.0150	**0.7626**±0.0030	*	**0.9413**±0.0028	**0.8524**±0.0340	**0.9155**±0.0115
FF	0.5159±0.0237	1.1062±0.0385	0.4895±0.0159	0.4960±0.0049	0.5124±0.0450	0.9225±0.0146	0.4487±0.0611
SM	0.9022±0.0055	0.5584±0.0133	0.7321±0.0027	0.5741±0.0103	0.9284±0.0043	0.9097±0.0129	0.8626±0.0136
TabPC	**0.9162**±0.0026	0.6558±0.0278	0.7442±0.0045	**0.6873**±0.0037	0.9324±0.0027	0.8864±0.0099	0.8970±0.0038

Table 12: **TABPC offers competitive performance with *training times* one or two orders of magnitude faster than SotA.** Even early, less performant models treated as a simple baseline like CTGAN and TVAE take longer to train than all circuit-based methods while achieving significantly worse results. No training times are recorded for TABPFN as it is a pre-trained foundation model.

Method	**Adult**		**Beijing**		**Default**		**Diabetes**		**Magic**		**News**		**Shoppers**		**Avg. Rank**
CTGAN	3798.4±173.8	(6)	4126.4±315.6	(6)	3836.4±235.4	(6)	13872.4±621.6	(6)	1797.2±160.7	(5)	8221.2±761.1	(7)	1453.0±68.6	(5)	5.86
TVAE	1309.6±102.3	(4)	1482.4±68.8	(4)	1705.0±102.2	(4)	3863.8±194.3	(4)	834.6±85.1	(4)	4588.0±125.7	(5)	579.2±72.7	(4)	4.14
GReaT	16714.2±573.1	(9)	13739.2±371.1	(8)	26138.4±691.2	(10)	20472.0±0.0000	(7)	6808.0±244.4	(8)	63508.5±39867.5	(10)	5238.4±2630.3	(8)	8.57
STaSy	6526.5±263.0	(7)	7034.0±299.6	(7)	6300.4±758.6	(7)	21701.6±605.7	(8)	4949.2±355.9	(7)	7079.8±693.4	(6)	4385.0±402.8	(7)	7.00
CoDi	25186.2±593.2	(10)	27023.8±396.6	(10)	20233.8±314.5	(9)	1027557.0±0.0000	(10)	9617.8±112.0	(9)	20495.5±394.9	(9)	8238.2±242.1	(9)	9.43
TabSyn	3082.5±516.0	(5)	3321.2±327.2	(5)	3438.5±578.1	(5)	6506.2±389.9	(5)	2270.8±369.8	(6)	3746.5±395.9	(4)	2441.2±192.9	(6)	5.14
TabDiff	11604.6±2638.3	(8)	14418.4±3527.0	(9)	13899.0±2724.0	(8)	32069.2±2849.5	(9)	11643.4±3487.7	(10)	16040.0±2865.3	(8)	10320.8±2256.3	(10)	8.86
FF	1.1±0.3	(1)	1.7±0.4	(1)	2.5±0.3	(1)	2.4±0.4	(1)	0.5±0.0	(1)	7.1±0.8	(1)	0.5±0.0	(1)	**1.00**
SM	59.7±6.9	(2)	106.3±5.8	(2)	163.9±4.3	(3)	293.4±7.8	(3)	15.4±0.7	(2)	268.4±41.3	(2)	47.6±0.8	(2)	2.29
TabPC	272.5±18.8	(3)	581.6±40.3	(3)	78.3±16.2	(2)	247.9±22.3	(2)	38.9±1.9	(3)	404.0±232.2	(3)	50.6±0.8	(3)	2.71

Table 13: **TABPC is the best ranked SotA method in terms of dataset *sampling time* (i.e. time taken to generate one dataset),** beating TABSYN and TABDIFF in six out of the seven datasets. In each case, the generated dataset is of the same size as the training dataset. Note the extremely high sampling time for TABPFN, which must autoregressively sample features and average over multiple feature ordering permutations for robustness.

Method	**Adult**		**Beijing**		**Default**		**Diabetes**		**Magic**		**News**		**Shoppers**		**Avg. Rank**
CTGAN	1.9±0.8	(4)	2.4±0.7	(4)	2.4±0.3	(5)	6.3±0.4	(4)	1.1±0.2	(5)	4.1±0.2	(3)	1.1±0.3	(5)	4.29
TVAE	1.2±0.4	(2)	1.9±0.8	(3)	1.9±0.4	(4)	6.1±2.8	(3)	0.7±0.2	(2)	4.0±2.1	(2)	0.6±0.1	(2)	2.57
GReaT	354.0±21.5	(10)	232.4±1.5	(10)	534.2±5.8	(10)	*	(10)	104.0±2.3	(10)	*	(11)	156.2±104.1	(10)	10.14
STaSy	48.9±75.9	(9)	13.4±0.5	(9)	16.9±11.1	(9)	5081.2±0.0000	(9)	5.8±3.6	(9)	18.9±8.2	(9)	7.0±3.4	(9)	9.00
CoDi	8.8±0.2	(7)	9.2±0.3	(8)	6.9±0.2	(7)	635.4±0.0000	(8)	3.5±0.2	(7)	8.3±0.3	(6)	3.0±0.2	(7)	7.14
TabSyn	5.0±0.2	(5)	5.4±0.4	(6)	5.2±0.6	(6)	15.6±1.7	(6)	2.6±0.2	(6)	11.1±1.7	(7)	2.0±0.2	(6)	6.00
TabDiff	10.8±0.5	(8)	9.0±0.4	(7)	9.6±0.3	(8)	176.5±1.4	(7)	4.4±0.4	(8)	16.2±0.4	(8)	6.0±0.5	(8)	7.71
TabPFN	571.3±1.6	(11)	623.6±9.3	(11)	1213.1±37.1	(11)	*	(10)	343.4±16.8	(11)	3978.5±60.9	(10)	254.1±3.1	(11)	10.71
FF	0.5±0.2	(1)	0.6±0.1	(1)	0.9±0.1	(1)	0.8±0.1	(1)	0.1±0.0	(1)	2.5±0.2	(1)	0.2±0.0	(1)	**1.00**
SM	1.3±0.0	(3)	1.4±0.1	(2)	1.8±0.0	(3)	3.1±0.0	(2)	1.0±0.0	(4)	4.4±0.1	(4)	1.0±0.0	(3)	3.00
TabPC	5.9±0.2	(6)	4.3±0.0	(5)	1.5±0.0	(2)	7.7±0.0	(5)	0.7±0.0	(3)	6.6±0.1	(5)	1.1±0.0	(4)	4.29

Table 14: Parameter counts for all methods and datasets. For TABPC, our choice of overparameterisation yields PCs that can have even an order of magnitude higher number of parameters compared to other DGMs. This observation is explained by PCs' tractability: since the only non-linearities are in the inputs, to attain similar expressivity to non-tractable DGMs, PCs must compensate by having more parameters. This does not come at the cost of an increased training time. TABPFN is omitted since it is a pre-trained foundation model.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers
CTGAN	9.60×10^6	9.58×10^6	9.66×10^6	1.18×10^7	9.56×10^6	9.89×10^6	9.60×10^6
TVAE	1.07×10^7	1.06×10^7	1.07×10^7	1.28×10^7	1.06×10^7	1.09×10^7	1.07×10^7
GReaT	1.21×10^8	1.21×10^8	1.21×10^8	*	1.21×10^8	*	1.21×10^8
STaSy	4.26×10^7	4.15×10^7	4.19×10^7	1.49×10^8	3.85×10^7	4.05×10^7	4.12×10^7
CoDi	1.19×10^7	1.19×10^7	1.19×10^7	1.19×10^7	1.19×10^7	1.19×10^7	1.19×10^7
TabSyn	1.06×10^7	1.06×10^7	1.07×10^7	1.08×10^7	1.06×10^7	1.09×10^7	1.07×10^7
TabDiff	2.12×10^7	2.12×10^7	2.14×10^7	2.16×10^7	2.12×10^7	2.18×10^7	2.13×10^7
FF	1.34×10^2	1.53×10^2	1.19×10^2	2.35×10^3	2.40×10^1	3.60×10^2	1.45×10^2
SM	2.70×10^6	7.70×10^6	2.40×10^6	4.69×10^7	2.50×10^5	3.61×10^6	2.92×10^6
TabPC	4.54×10^8	3.19×10^8	1.29×10^7	2.31×10^8	6.30×10^7	1.23×10^8	1.51×10^8

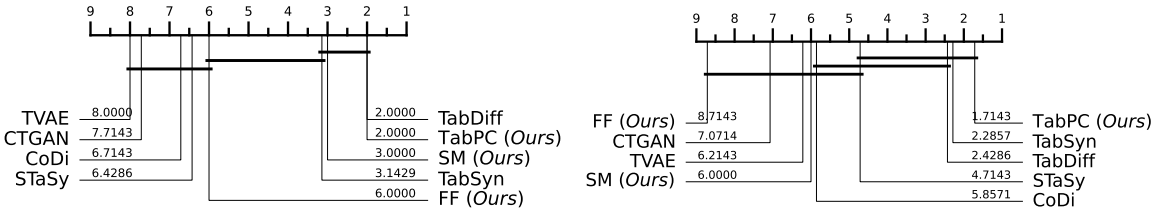


Figure 11: **TABPC falls in the top clique of both CDDs**, and it is clear that **WNMIS (right) is able to successfully identify the trivial FF model** where TREND (left) is not able to.

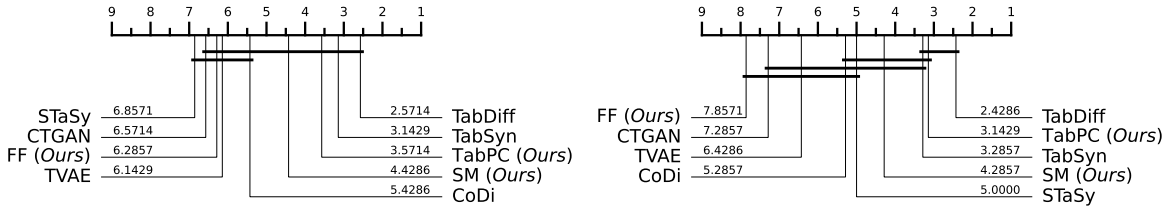


Figure 12: **TABPC is in the top clique for both α-PRECISION (left) and β-RECALL (right).** This suggests that its synthetic samples are competitive with the fidelity and diversity of samples generated by TABDIFF and TABSYN.

It is important to note that this implementation requires observations for all datasets. Therefore, methods which cannot be trained or generate successfully, or whose metric computations fail across some datasets (such as GREAT and STASY) will not appear in the corresponding CDD. For this reason, since TABPFN did not work for Diabetes, it does not appear in any CDDs.

E.3 SHALLOW MIXTURE RESULTS

RQ) How much does increasing PC complexity help? That is, how close to TDG SotA can we get with just a shallow mixture?

The SM models have one to two orders of magnitude fewer parameters than TABPC, and typically one order fewer than the diffusion-based SotA (except on Diabetes). For this reason, they represent a good intermediate level of complexity between FF and TABPC— expressive enough to increase their performance significantly, with a more comparable parameter count to other models. Recall that, as tractable models, PCs require more parameters in order to compensate for having

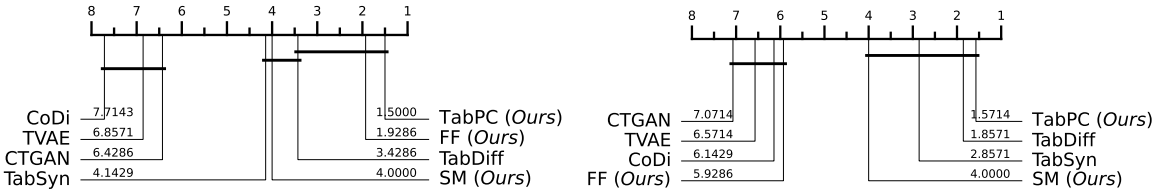


Figure 13: **C2ST (XGB) (right) is able to clearly identify the trivial FF model** where the metric using LR (left) is not. All lower performing models do poorly on C2ST (XGB) and belong in the same clique, whereas the diffusion-based SotA, TabPC and SM fall within the top clique and hence have comparable performance.

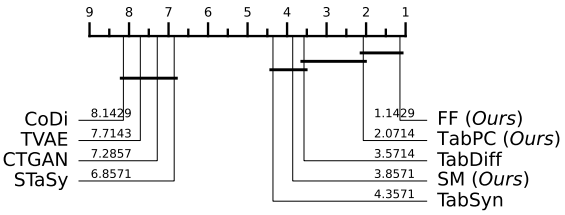


Figure 14: **TabPC and FF form the top clique for Shape,** showing that they accurately recover the univariate marginals. This is somewhat expected for FF due to the pre-processing and the fact that we fit all marginals. The lower performing models again form a bottom clique (CoDi, TVAE, CTGAN, STaSy).

non-linearities only in their inputs.

It is clear from the results that the SM models are a tier below TabPC and the diffusion-based SotA. In the CDDs of App. E.2, they often appear in fourth place after these models. In numerical terms, such as on C2ST (XGB), while not generally competitive with SotA models they offer a significant step up from FF, even beating TabPC on Beijing. Where FF posts results which are very close to zero, SM make a step in the direction of SotA performance.

Their training times are also typically faster than for TabPC, as to be expected, but for some datasets it can be lower (in particular, Default, where the SotA version of TabPC is smaller than for other datasets and hence ends up being faster).

Given that simply mixing together many FF distributions is a naive way of constructing a generative model, it is interesting that we can achieve reasonable results with the use of SM.

E.4 UTILITY METRIC RESULTS

MLE is designed to measure how useful synthetic data is for the purpose of training new machine learning models. This focuses on synthetic data's use as a proxy for real data. Here, we follow the protocol and implementation of Shi et al. 2025.

To compute MLE for a given dataset, we train a discriminative machine learning model (in this case, an XGBoost regressor / classifier depending on the dataset task) on that dataset, and evaluate it on hold-out test data from the real distribution.

As mentioned, we follow the same protocol as Shi et al. 2025, and reuse their implementation. Specifically, we split the given dataset into training and validation sets with a ratio of $8:1$, learn the discriminative model on this training set, and use this validation set to select the optimal hyperparameters. Then the performance on the test data is evaluated, which is the value reported for MLE. It is important to compare this value against the value obtained by using the 'real' training data, which can be seen in the row 'Real' in Tab. 11. As seen in the table, training with real data yields the best results across all datasets, but some results for SotA models come close to matching this performance.

We observe that the data generated by TabPC is reasonably competitive with SotA models. Downstream discriminative models trained on data from TabPC sometimes outperform that of SotA diffusion models, and are sometimes outperformed. We note, as do previous works [Shi et al., 2025], that methods which typically generate data of low fidelity are sometimes able to achieve good scores, so this is not the most convincing metric. However, it provides some indication of how useful

the generated data could be for downstream ML tasks.

TabPFN generates synthetic data which performs well under this metric, generally comparable or better to SotA models. This is perhaps not entirely surprising given its connection to predictive tasks, but is interesting in the context that its fidelity scores are generally lower than these models.

We strongly remark that one practice to avoid when reporting MLE is averaging over the numerical results, as is done in Guzmán-Cordero et al. 2025. Since the objective for each dataset is different (i.e. we want to minimise RMSE for regression tasks, and maximise AUROC for classification tasks), it does not make sense to average in this way as the two metrics are "incomparable" [Javaloy et al., 2025].

E.5 PRIVACY METRIC RESULTS

It is complex to evaluate the privacy protection of synthetic data. Several recent works [Zhang et al., 2024, Shi et al., 2025, Guzmán-Cordero et al., 2025] rely on a simple distance-based metric called DCR (Distance to Closest Record).

As a privacy metric, distance to closest record (DCR) is motivated by the idea that if synthetic data and training data are too close, according to some distance metric, then there may be some information leakage.

The original DCR score, as used for example in Shi et al. 2025, Zhang et al. 2024, represents the probability that a generated data sample is closer to the training data than to some hold-out test set. In this setup, values closer to 50% are desirable, as they indicate more even distance between the synthetic data and both the training and hold-out sets.

One issue with implementing this is that it requires retraining models with a modified data split (50/50 train and test, so that the training and hold-out sets are balanced). As another issue, ***using DCR as a proxy metric for privacy has recently been strongly criticised and described as "flawed by design" in Yao et al. 2025, which recommends moving away from DCR***. For this reason, we report these results here in Tab. 15 for comparison only.

Here, we also compute an alternative formulation to previous works [Shi et al., 2025, Zhang et al., 2024]. This is because their original formulation requires splitting the training data in half and retraining a new model. (Results for the original DCR formulation are also reported for reference in Tab. 15, excluding TabPFN for which this metric was not computed due to the different data split.)

We instead use a formulation from Palacios et al. 2025 which compares the distributions of distances from the synthetic data and a test set to the training data. If there are significantly more synthetic entries closer to the training data than expected (i.e. compared to the test data), then this indicates a potential privacy risk.

We call this metric 'quantile DCR', and denote it by DCR-002 and DCR-005 in the following tables, with the number corresponding to the quantile (either 2% or 5% respectively). We observe the corresponding results for quantile DCR in Tab. 16 and Tab. 17. No privacy leaks for TabPC are suggested according to the DCR-002 and DCR-005 metrics, which compare the sizes of the 2% and 5% quantiles of distances respectively. In this respect, TabPC offers a similar risk of privacy leaks as TabDiff and TabSyn, which also show no indication of privacy leaks under these metrics.

Table 15: According to DCR, TabPC offers a similar privacy risk as SotA diffusion-based models. However, we recall the criticisms of Yao et al. 2025 that DCR is a "flawed by design" metric. These numbers are reported for reference to compare with older works. For DCR, values closer to 0.5 are preferable.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers
CTGAN	0.4838±0.0184	0.4969±0.0149	0.4970±0.0047	0.5041±0.0047	0.4972±0.0060	0.4944±0.0014	0.4934±0.0562
TVAE	0.5027±0.0119	0.4892±0.0145	0.5024±0.0046	0.3618±0.2028	0.4958±0.0021	0.5104±0.0069	0.4780±0.0304
GReaT	0.5194±0.0042	0.5124±0.0040	0.4877±0.0028	*	0.5131±0.0045	*	0.4934±0.0046
STaSy	0.5020±0.0052	0.5047±0.0030	0.5061±0.0048	0.9974±0.0000	0.5103±0.0053	0.5029±0.0104	0.4866±0.0329
CoDi	0.5021±0.0055	0.5057±0.0061	0.5053±0.0066	0.5048±0.0000	0.5180±0.0035	0.4976±0.0021	0.5057±0.0060
TabSyn	0.5069±0.0012	0.5022±0.0016	0.5084±0.0020	0.5071±0.0018	0.5045±0.0029	0.4953±0.0021	0.5069±0.0044
TabDiff	0.5488±0.0071	0.5139±0.0025	0.5221±0.0021	0.5181±0.0079	0.5011±0.0032	0.5091±0.0026	0.5120±0.0028
FF	0.5034±0.0011	0.4978±0.0011	0.5127±0.0018	0.5057±0.0028	0.4876±0.0024	0.4852±0.0024	0.5021±0.0043
SM	0.5498±0.0014	0.5545±0.0041	0.5133±0.0025	0.5067±0.0022	0.5348±0.0023	0.4942±0.0025	0.5091±0.0048
TabPC	0.5431±0.0047	0.5162±0.0020	0.5072±0.0031	0.5208±0.0025	0.5295±0.0028	0.4943±0.0025	0.5128±0.0024

However, we stress again that, according to Yao et al. 2025, distance-based privacy metrics do not even give a strong

Table 16: According to DCR-002, TABPC may offer a similar risk of privacy leakage as SotA diffusion-based models. For DCR-002, numbers equal to or lower than 2 may suggest a lower risk of privacy leakage.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers
CTGAN	0.0934±0.1242	<0.0001	<0.0001	0.0226±0.0096	0.0023±0.0052	0.0824±0.0175	<0.0001
TVAE	5.7775±1.6151	0.0059±0.0029	0.0874±0.0350	14.1846±9.6827	0.0140±0.0147	0.2175±0.0697	0.0162±0.0075
GReaT	<0.0001	1.4199±0.0606	12.5178±0.1007	*	0.3529±0.0517	*	10.6047±0.2320
STaSy	0.9809±0.2132	0.4454±0.1188	0.4504±0.1140	<0.0001	0.2746±0.1323	0.2433±0.1508	0.7101±0.9099
CoDi	0.0276±0.0104	<0.0001	<0.0001	<0.0001	0.0374±0.0158	0.0006±0.0013	<0.0001
TabSyn	1.6842±0.1084	0.5732±0.1172	1.1444±0.0626	0.6296±0.0252	0.0584±0.0160	1.0325±0.1639	1.8564±0.2962
TabDiff	1.8673±0.0624	0.6386±0.0518	1.4067±0.0647	1.0662±0.5802	0.0608±0.0270	0.8178±0.1928	2.2601±0.2484
TabPFN	2.4932±0.0595	2.6370±0.0725	0.8437±0.0848	*	0.2059±0.0250	0.2522±0.0040	0.0180±0.0169
FF	0.0424±0.0120	0.0841±0.0101	<0.0001	0.0111±0.0036	<0.0001	<0.0001	0.0018±0.0040
SM	2.0159±0.1441	0.9340±0.0701	0.8207±0.1759	0.0694±0.0398	0.5398±0.1367	0.0695±0.0597	1.6419±0.3882
TabPC	1.9029±0.1342	0.5311±0.0419	0.8719±0.0937	1.4344±0.0495	0.1718±0.0479	0.4316±0.3200	2.5773±0.1746

Table 17: According to DCR-005, TABPC may offer a similar risk of privacy leakage as SotA diffusion-based models. For DCR-005, numbers equal to or lower than 5 may suggest a lower risk of privacy leakage.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers
CTGAN	0.3225±0.4189	0.0117±0.0058	0.0222±0.0191	0.1209±0.0437	0.1122±0.0263	0.4686±0.0401	1.2183±0.4743
TVAE	15.0886±4.1617	0.0804±0.0051	1.2956±0.3023	31.5980±18.4872	0.7420±0.1228	1.5768±0.4036	10.2550±3.2313
GReaT	<0.0001	3.5747±0.1059	23.9711±0.2405	*	4.8373±0.0413	*	21.8780±0.3721
STaSy	2.5134±0.4673	1.3539±0.3471	1.6000±0.2372	<0.0001	2.8533±0.4710	0.9827±0.4453	2.4403±1.6136
CoDi	0.1480±0.0270	0.0037±0.0040	<0.0001	<0.0001	1.5096±0.2736	0.0241±0.0117	0.6759±0.3872
TabSyn	4.3039±0.2098	1.6109±0.3337	3.1133±0.1441	2.0344±0.0849	1.5283±0.1844	2.6268±0.3070	4.9941±0.4275
TabDiff	4.6829±0.1926	1.7179±0.1245	3.9593±0.0926	2.8962±1.5304	1.5377±0.2004	2.0561±0.5072	5.9187±0.9103
TabPFN	6.3039±0.1552	6.6539±0.1011	4.0415±0.2526	*	2.8013±0.0721	1.5920±0.0277	2.5088±0.1153
FF	0.1112±0.0250	0.2411±0.0115	<0.0001	0.0671±0.0140	0.0023±0.0032	<0.0001	0.2145±0.0443
SM	5.0293±0.1940	2.3703±0.1427	2.3252±0.4999	0.3331±0.1147	5.6938±0.6394	0.3610±0.1499	4.4174±0.5258
TabPC	4.8438±0.2155	1.4757±0.1191	2.7200±0.1932	3.9087±0.1079	3.0309±0.3601	1.2136±0.6237	6.2720±0.1197

Table 18: TABPC BPD values (see Eq. (20)) on all datasets and splits (using the hyperparameters in App. D.5)

Dataset	TABPC BPD		
	Training	Validation	Test
Adult	1.1227	1.0476	1.0500
Beijing	1.3615	1.3582	1.4655
Default	0.3431	0.3458	0.5583
Diabetes	1.2901	1.1650	1.2906
Magic	0.5294	0.5283	0.9391
News	0.3894	0.3884	0.5832
Shoppers	0.9056	0.8978	1.0208

indication of dataset privacy — stronger conclusions would require a much more involved and rigorous investigation. Since our primary aim is to highlight the performance of TABPC as a fast and high-fidelity TDG method, we leave such an investigation to future work.

F LIKELIHOOD VS C2ST (XGB) EXPERIMENT

F.1 BITS-PER-DIMENSION (BPD)

Bits-per-dimension offers a log-likelihood normalised by the number of features, making values more comparable across datasets. It represents the average number of bits required to encode a single dimension.

For a dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ consisting of datapoints $\boldsymbol{x} \in X_1 \times \ldots \times X_D$, we define the BPD by

$$\mathrm{BPD}(\mathcal{D}) = \frac{\mathrm{NLL}(\mathcal{D})/N}{\log 2 \cdot D}, \tag{20}$$

where $\mathrm{NLL}(\mathcal{D})$ denotes the negative log-likelihood of dataset $\mathcal{D}$ under the given model.

F.2 LIKELIHOOD VS C2ST (XGB) PLOTS

Detailed plots for all datasets can be found in Fig. 15. As mentioned previously, each marker in the plot corresponds to a different hyperparameter configuration. These hyperparameters are number of (sum and input) units, batch size, and learning rate. Number of units are selected to be powers of 2. The minimal grid for each dataset tests: (i) number of units 128, 512, 2048; (ii) batch size 64, 256, 512 (iii) learning rates 0.1, 0.25, and 0.5. Where possible, we also test higher numbers of units. Early iterations tested lower learning rates (0.01, 0.001) which proved to be much less effective, and so these do not appear for all datasets.

Regression lines are fit with Huber loss using the `HuberRegressor` implementation of `scikit-learn`. Further details can be found at https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.HuberRegressor.html.

F.3 BPD TABLES

Tab. 18 displays BPD values across all datasets and splits.

F.4 NUMBER OF UNITS K VS C2ST (XGB) PLOTS

To better highlight the relationship between the number of units used in TABPC K and the downstream C2ST (XGB), here we provide additional plots studying this, which can be found in Fig. 16. Note that these simply replot the information in Fig. 15 with K as the x-axis, but it helps to visualise the effect.

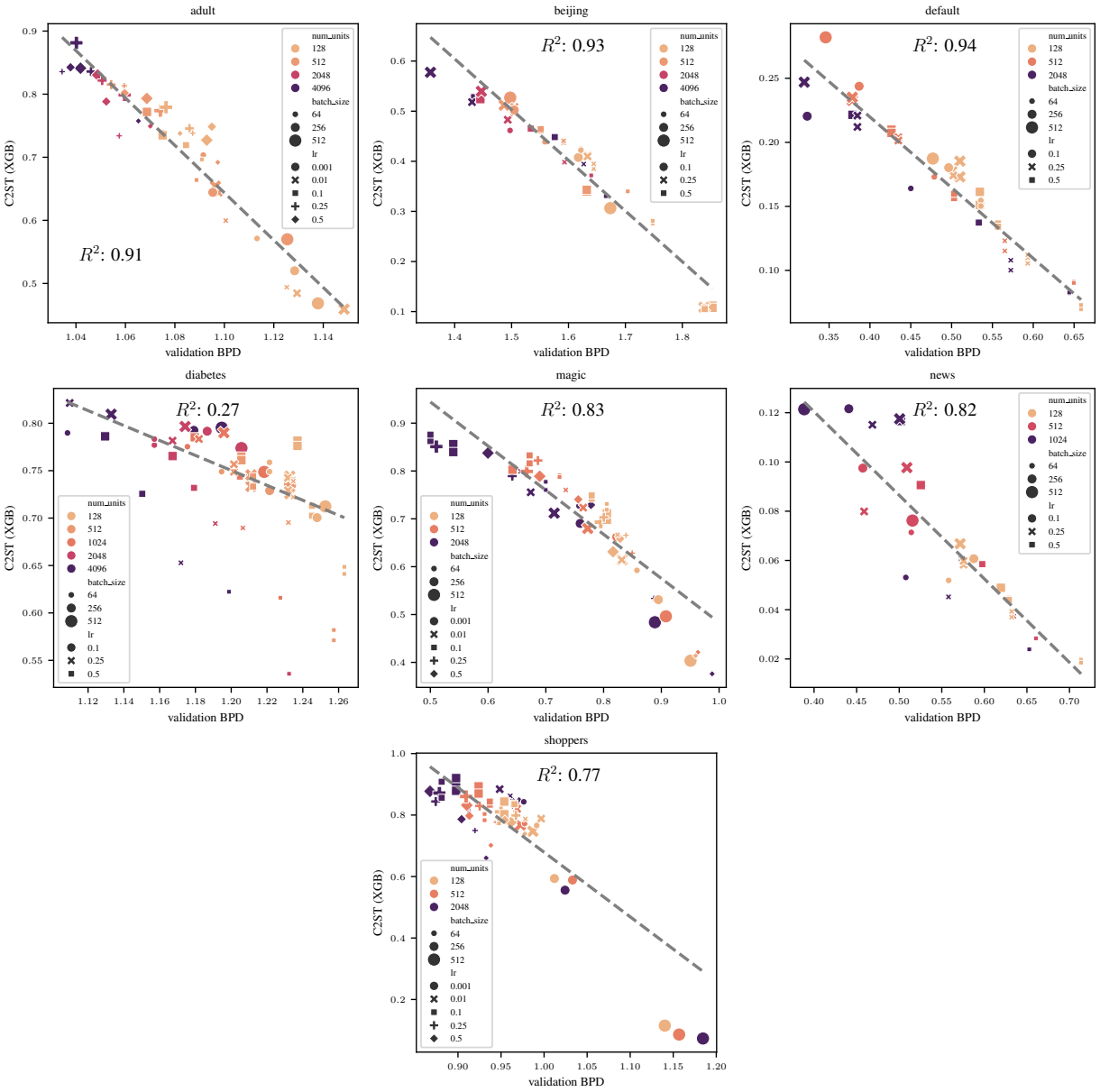


Figure 15: **Validation set bits-per-dimension (BPD) and downstream sample quality are correlated** across all datasets. In the image domain, PCs typically struggle to generate high-quality image samples while achieving low BPDs, and so we highlight this correlation here in the tabular domain.

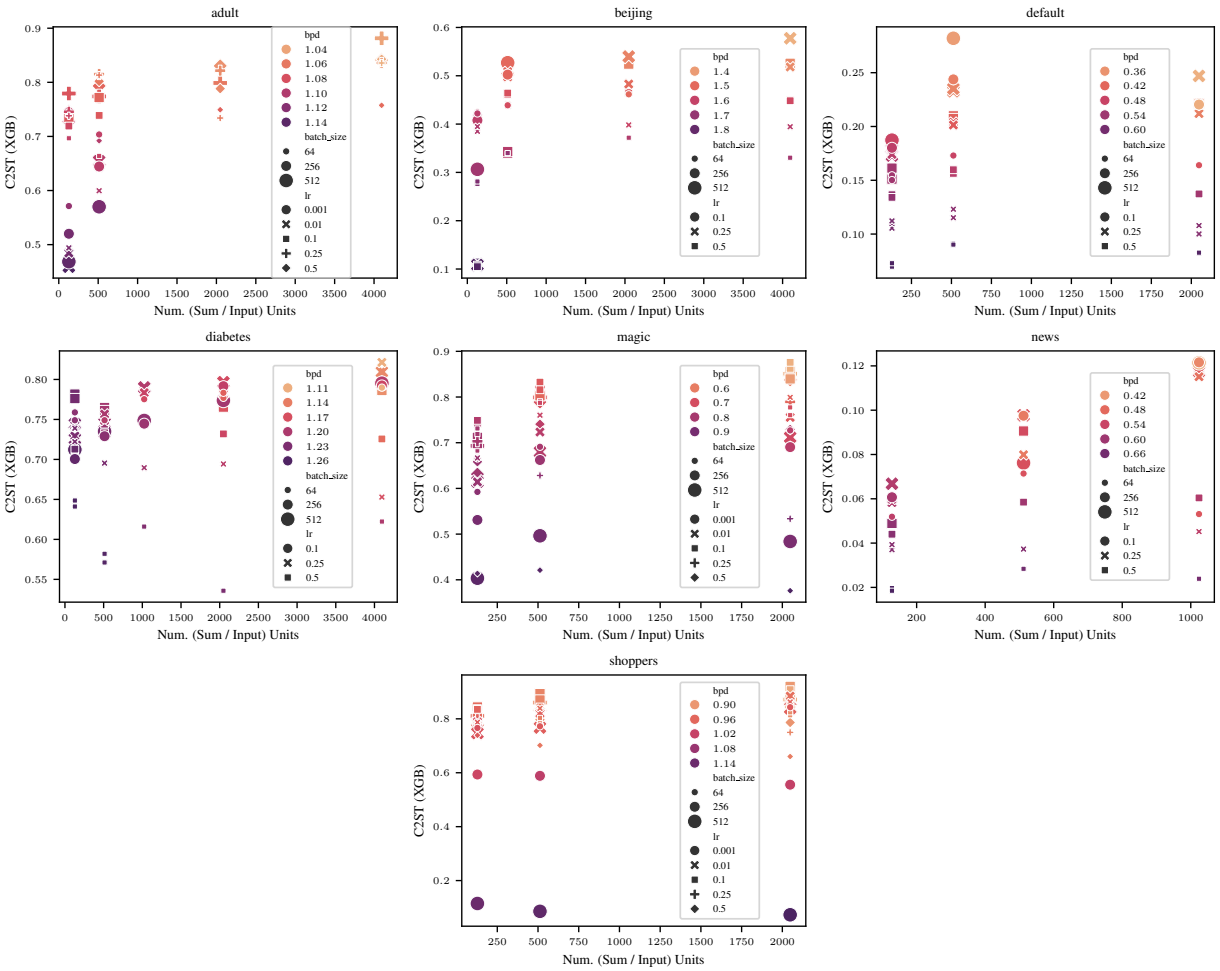


Figure 16: **In general, increasing the number of sum and input units K corresponds to increased C2ST (XGB), as expected.** Quoted BPD values are for the validation set. This replots the data from Fig. 15 to focus on the effect of increasing K.

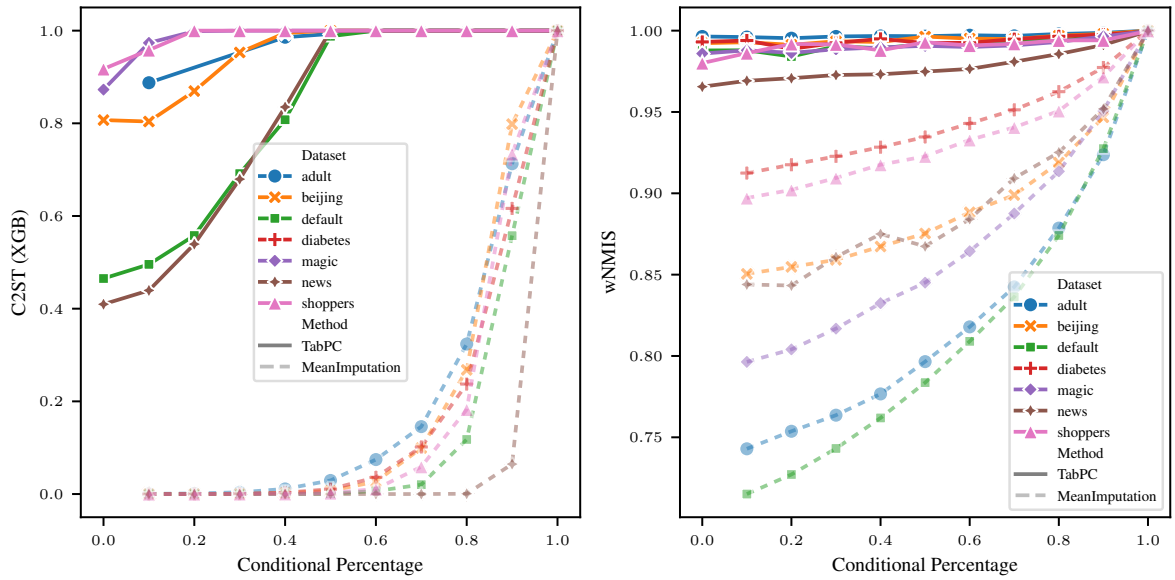


Figure 17: **TabPC can realistically impute unobserved features when conditioned on any features**, and it can do this exactly and efficiently without retraining as a tractable probabilistic model. Realism is measured by C2ST (XGB). The x-axis denotes the percentage of values on which we condition to generate the samples; 0% conditioning corresponds to unconditional generation, up to 100% conditioning which corresponds to copying the observed data. Dashed translucent lines denote the mean (or mode for categorical features) imputation baseline, which imputes missing values based on the observed column mean (or mode). By injecting increasing amounts of information about the training set via conditioning, we can generate highly realistic samples which can almost perfectly fool the XGBoost classifier.

G CONDITIONAL SAMPLING EXPERIMENT

Full-sized plots for this experiment when conditioning on the test set are found in Fig. 17.

H EVALUATION OF PROMOTED METRICS

H.1 WNMIS VS TREND

Critical difference diagrams (CDDs) for the two metrics can be seen in Fig. 11. We note first that, despite the FF model being of average rank 6.0 for Trend, it is in the same clique as the recent diffusion-based model TabSyn, which tends to be one of the top performing models. The CDD for wNMIS instead shows that FF is relegated to last place, as should be expected since it cannot model any correlations.

Aside from this, both metrics assign methods to reasonable cliques – the most recent and performant methods, TabDiff and TabSyn form the top clique, and the more simplistic or earlier models, CTGAN, TVAE, STaSy and CoDi, are in a lower clique. Moreover, the diffusion-based approaches (STaSy and CoDi) can form a bridge between the lower clique and upper clique (meaning that they can be connected to methods from both), whereas the GAN or VAE-based methods do not, and remain firmly in the lower clique. This is reasonable since these GAN / VAE-methods are now the oldest.

Full observations for each metric can be seen in Tab. 5 and Tab. 6.

We also note that, at the top end of performance, we achieve scores which are very close to 1.0. wNMIS therefore does not alleviate the issue of metric saturation.

However, wNMIS provides a more principled approach overall to quantifying how well synthetic data models correlations. This is shown by its ability to separate out the simplistic FF baseline, whereas Trend assigns it almost perfect scores.

H.2 C2ST (XGB) VS C2ST (LR)

Using XGBoost as our classifier for C2ST, we can now clearly separate out the FF model from the top-performing models. This can be seen in both Fig. 1 and Fig. 13. In fact, in terms of C2ST (LR), the simplistic FF model even beats the sophisticated diffusion-based TABDIFF and TABSYN, likely because it is designed to direct match the empirical means. This provides some empirical support to our theoretical result in App. A, and underscores the insufficiency of C2ST (LR) for the evaluation of synthetic data fidelity.

Relative to older baselines, TABSYN and TABDIFF have made some progress in being able to fool XGBoost across most datasets. However, the News dataset, which is the most difficult to model, is still lacking across all methods. Moreover, no values are as close to 1.0 as with LR, suggesting that there is still more to be done in improving synthetic data fidelity.

H.3 WNMIS VS NMIS

In this section, we investigate the advantages of *weighting* the normalised mutual information similarities.

The unweighted NMIS is defined as

$$\text{NMIS} := \frac{1}{\#\{x_i, x_j \,:\, i < j\}} \cdot \sum_{\{x_i, x_j \,:\, i<j\}} \text{NMIS}(x_i, x_j) \in [0, 1], \tag{21}$$

where $\text{NMIS}(x_i, x_j)$ is identically defined as in Eq. (4). Each pair therefore contributes equally to the score, as opposed to Eq. (4) which emphasises the contribution of pairs with high NMI in the real data and / or high NMI in the synthetic data. The motivation of this is that we want the score to better represent how well the model's generated data captures inter-pair dependencies *where they exist*; in principle, by having many independent feature pairs, the score could otherwise be pulled up.

Results for unweighted NMIS can be found in Tab. 19. One observation is that all values for NMIS are significantly higher than for WNMIS, making it more difficult to distinguish between model performances. Moreover, the weighting provides a more principled approach to penalising errors *only where dependencies exist*, as opposed to being affected by the presence of independent columns. It is for these reasons that we use WNMIS and not NMIS in the main text.

We note that the `SDMetrics` library has concurrently been updated to include a threshold in the TREND score computation (https://github.com/sdv-dev/SDMetrics/releases/tag/v0.27.0). That is, column pair scores are only included in the average if the association metric (either correlation or Cramer's V) is above this threshold. This helps to mitigate some of the observed issues we note in the main paper. However, we note several issues with this.

- These thresholds are rather arbitrarily set.[8] The weighting in WNMIS provides a more principled approach. Moreover, the weighting also results in the score decreasing when we have synthetic correlations but not real correlations between a pair of features. This is also an error in the synthetic data, but would be obfuscated by applying this thresholding technique (since then the error is not included in the trend average).
- Correlation and contingency similarities are not necessarily commensurable. Simply averaging over all column pair scores means that overall scores may vary based solely on the relative proportions of continuous, discrete, and mixed pairs. For WNMIS in contrast, by estimating the mutual information for feature pairs, we make scores commensurable, and so scores should be more comparable across different proportions of feature pair types.

H.4 ROBUSTNESS OF WNMIS TO DISCRETISATION

In this section we consider the sensitivity of WNMIS to the number of bins used to discretise the continuous feature(s). To do this, we compute the WNMIS score of TABPC for each dataset using a different number of bins to discretise continuous features. Results for this ablation can be found in Tab. 20.

[8] See https://github.com/sdv-dev/SDMetrics/blob/7b5c44d4576ca4ff939fb221e32c8ae43d2946db/sdmetrics/reports/single_table/quality_report.py#L16-L17.

Table 19: NMIS results across all methods **show more saturated scores than wNMIS**, since all values are above 0.9, whereas for wNMIS, the weakest methods posted lower results. For a clear example of this, compare the FF results here (where the lowest score is 0.8756 on Magic) to their wNMIS scores (where the lowest is 0.6942 on Default).

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.9737±0.0035 (8)	0.9667±0.0097 (9)	0.9537±0.0050 (10)	0.9930±0.0004 (5)	0.9594±0.0037 (10)	0.9848±0.0007 (9)	0.9615±0.0079 (11)	8.86
TVAE	0.9720±0.0023 (9)	0.9305±0.0137 (11)	0.9794±0.0021 (7)	0.9813±0.0081 (9)	0.9805±0.0022 (6)	0.9863±0.0006 (7)	0.9832±0.0017 (8)	8.14
GReaT	0.9889±0.0001 (7)	0.9946±0.0002 (5)	0.9617±0.0005 (9)	* (10)	0.9776±0.0005 (8)	* (11)	0.9896±0.0006 (5)	7.86
STaSy	0.9903±0.0010 (6)	0.9929±0.0017 (6)	0.9876±0.0037 (5)	0.9907±0.0000 (8)	0.9840±0.0087 (5)	0.9930±0.0010 (3)	0.9909±0.0005 (4)	5.29
CoDi	0.9648±0.0008 (10)	0.9915±0.0031 (8)	0.9635±0.0031 (8)	0.9927±0.0000 (6)	0.9795±0.0021 (7)	0.9900±0.0005 (6)	0.9773±0.0023 (9)	7.71
TabSyn	0.9949±0.0006 (4)	0.9957±0.0007 (4)	0.9955±0.0005 (2)	0.9974±0.0001 (2)	0.9975±0.0002 (1)	0.9952±0.0010 (1)	0.9958±0.0006 (3)	2.43
TabDiff	0.9965±0.0002 (2)	0.9978±0.0001 (2)	0.9967±0.0003 (1)	0.9970±0.0024 (3)	0.9965±0.0009 (2)	0.9920±0.0025 (4)	0.9965±0.0001 (2)	2.29
TabPFN	0.9959±0.0002 (3)	0.9987±0.0001 (1)	0.9863±0.0004 (6)	* (10)	0.9946±0.0006 (4)	0.9904±0.0002 (5)	0.9894±0.0002 (6)	5.00
FF	0.9594±0.0000 (11)	0.9637±0.0000 (10)	0.9339±0.0000 (11)	0.9924±0.0000 (7)	0.8756±0.0000 (11)	0.9791±0.0000 (10)	0.9770±0.0000 (10)	10.00
SM	0.9930±0.0005 (5)	0.9927±0.0005 (7)	0.9901±0.0006 (4)	0.9934±0.0008 (4)	0.9699±0.0021 (9)	0.9853±0.0006 (8)	0.9865±0.0007 (7)	6.29
TabPC	0.9979±0.0004 (1)	0.9973±0.0002 (3)	0.9949±0.0003 (3)	0.9984±0.0003 (1)	0.9955±0.0007 (3)	0.9940±0.0006 (2)	0.9975±0.0003 (1)	2.00

Table 20: **wNMIS is robust to the number of discretisation bins.** Using TabPC, for every considered dataset we find that the computation of wNMIS varies little with the number of bins chosen to discretise continuous features. Throughout the rest of the paper, we use 10 bins.

Num. Bins	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers
5	0.9949±0.0038	0.9924±0.0010	0.9909±0.0016	0.9919±0.0025	0.9915±0.0018	0.9231±0.0086	0.9896±0.0015
10	0.9933±0.0049	0.9925±0.0010	0.9894±0.0012	0.9919±0.0024	0.9925±0.0012	0.9388±0.0048	0.9927±0.0009
20	0.9924±0.0051	0.9924±0.0008	0.9872±0.0011	0.9921±0.0024	0.9929±0.0006	0.9459±0.0045	0.9949±0.0007
50	0.9923±0.0050	0.9932±0.0008	0.9854±0.0009	0.9923±0.0023	0.9939±0.0006	0.9512±0.0027	0.9958±0.0004

I α-PRECISION AND β-RECALL IMPLEMENTATION DISCREPANCY

After publication, we realised that our α-PRECISION and β-RECALL results were being computed using a "naive" implementation internal to the evaluation package Synthcity. This is the result of the evaluation code for these two metrics being passed down through several different codebases: first from the repository of GOGGLE, to that of TabSyn, and finally into TabDiff, from which we inherited our repository's structure and this particular piece of evaluation code.

Based on the Synthcity code, it appears that the main difference between this *"naive"* implementation and the *"original"* implementation described by Alaa et al. 2022 (which is the approach outlined in App. D.3.2) is that data-points are not embedded in a hypersphere in the way described there; instead, they are normalised via a min-max scaling to the interval $[0, 1]$. The comparisons between sets used to compute α-PRECISION and β-RECALL are then made in this normalised space, instead of the hypersphere.

We are unsure why this implementation was used originally in GOGGLE / TabSyn, and it seems that other people have also noticed this discrepancy in the past, without the intention behind this decision being made clear. It appears as though this issue of using the "naive" implementation has been replicated across multiple different works without being spotted (in particular, at least for the repositories linked above which are the most comparable recent works included as baselines).

We have therefore updated all figures and tables in the main paper to instead use the numbers computed under this "original" implementation, as opposed to the "naive" implementation. Importantly, this means that our numbers for these metrics are no longer backwards compatible with works which report numbers computed with the "naive" implementation, such as those mentioned above. For compatibility, we retain our previously computed metrics under new tables, Tab. 21 and Tab. 22. We also present their corresponding CDDs in Fig. 18.

We note in particular that the previous version of Fig. 1 displayed a very similar trend to the current version, with the exception that the α-PRECISION scores for the News dataset are now significantly lower (the same also holds for the β-RECALL scores for News).

Table 21: We present the "naive" implementation results for α-PRECISION for backwards compatibility with prior works. **TABPC is the best ranked and best performing model in terms of α-PRECISION (naive)**, a measure of sample realism, attaining scores greater than 0.99 in six out of seven datasets, and being ranked first for all datasets.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.7539±0.0362 (9)	0.9494±0.0365 (7)	0.6734±0.0486 (11)	0.7983±0.0178 (6)	0.9040±0.0200 (8)	0.9750±0.0071 (2)	0.7790±0.0374 (10)	7.57
TVAE	0.6233±0.0595 (10)	0.8622±0.0651 (10)	0.9087±0.0392 (7)	0.1084±0.0749 (8)	0.9684±0.0104 (5)	0.9216±0.0380 (7)	0.4797±0.1287 (11)	8.29
GReaT	0.5898±0.0010 (11)	* (11)	0.8657±0.0036 (8)	* (10)	0.8678±0.0049 (9)	* (11)	0.7843±0.0030 (9)	9.86
STaSy	0.8872±0.0809 (7)	0.9199±0.0612 (9)	0.9385±0.0282 (5)	<0.0001 (9)	0.9362±0.0501 (7)	0.9218±0.0548 (6)	0.9009±0.0694 (8)	7.29
CoDi	0.8021±0.0287 (8)	0.9736±0.0089 (5)	0.8189±0.0063 (9)	0.4335±0.0000 (7)	0.8627±0.0055 (10)	0.9159±0.0077 (9)	0.9186±0.0271 (7)	7.86
TabSyn	0.9902±0.0042 (3)	0.9822±0.0052 (3)	0.9889±0.0033 (2)	0.9795±0.0095 (4)	0.9938±0.0021 (3)	0.9558±0.0099 (4)	0.9898±0.0019 (2)	3.00
TabDiff	0.9861±0.0040 (4)	0.9779±0.0028 (4)	0.9873±0.0032 (3)	0.9422±0.0106 (5)	0.9939±0.0022 (2)	0.9172±0.0576 (8)	0.9851±0.0142 (3)	4.14
TabPFN	0.9037±0.0031 (6)	0.9946±0.0007 (2)	0.9287±0.0033 (6)	* (10)	0.9747±0.0043 (4)	0.9539±0.0009 (5)	0.9200±0.0059 (6)	5.57
FF	0.9568±0.0036 (5)	0.9531±0.0009 (6)	0.6969±0.0006 (10)	0.9816±0.0015 (3)	0.8612±0.0008 (11)	0.8378±0.0028 (10)	0.9713±0.0043 (4)	7.00
SM	0.9915±0.0023 (2)	0.9394±0.0102 (8)	0.9699±0.0048 (4)	0.9820±0.0085 (2)	0.9654±0.0055 (6)	0.9732±0.0061 (3)	0.9616±0.0231 (5)	4.29
TabPC	0.9961±0.0014 (1)	0.9951±0.0011 (1)	0.9950±0.0008 (1)	0.9953±0.0023 (1)	0.9948±0.0017 (1)	0.9824±0.0064 (1)	0.9917±0.0046 (1)	**1.00**

Table 22: We present the "naive" implementation results for β-RECALL for backwards compatibility with prior works. **TABPC has the second highest average rank in β-RECALL (naive)**, a measure of sample diversity. TABPC offers performance which generally competitive with SotA, but it sometimes falls slightly behind TABDIFF which is the highest ranked method for this metric. Note the relatively high scores of TABPFN, which suggests that it is able to generate a comparably diverse range of samples.

Method	Adult	Beijing	Default	Diabetes	Magic	News	Shoppers	Avg. Rank
CTGAN	0.1313±0.0613 (8)	0.3850±0.0203 (9)	0.1050±0.0124 (10)	0.0948±0.0050 (5)	0.1588±0.0076 (10)	0.2339±0.0130 (8)	0.2378±0.0197 (8)	8.29
TVAE	0.1221±0.0133 (9)	0.0540±0.0233 (10)	0.3039±0.0121 (8)	0.0150±0.0134 (7)	0.3631±0.0124 (9)	0.2811±0.0256 (7)	0.1909±0.0488 (11)	8.71
GReaT	0.4825±0.0001 (4)	* (11)	0.4211±0.0012 (5)	* (10)	0.3995±0.0041 (8)	* (11)	0.4551±0.0061 (5)	7.71
STaSy	0.3454±0.0170 (7)	0.4796±0.0348 (7)	0.3883±0.0185 (6)	<0.0001 (9)	0.4387±0.0272 (7)	0.3870±0.0147 (3)	0.3518±0.0537 (7)	6.57
CoDi	0.0875±0.0092 (10)	0.5377±0.0081 (5)	0.1863±0.0026 (9)	0.0138±0.0000 (8)	0.5100±0.0081 (1)	0.3662±0.0082 (4)	0.1915±0.0063 (10)	6.71
TabSyn	0.4796±0.0059 (5)	0.5221±0.0440 (6)	0.4598±0.0163 (3)	0.3521±0.0070 (3)	0.4775±0.0041 (6)	0.4310±0.0282 (2)	0.4785±0.0047 (3)	4.00
TabDiff	0.5255±0.0092 (1)	0.5981±0.0029 (3)	0.5154±0.0046 (1)	0.3690±0.1486 (2)	0.4799±0.0091 (5)	0.3561±0.0739 (5)	0.5107±0.0249 (1)	**2.57**
TabPFN	0.5133±0.0018 (2)	0.7340±0.0020 (1)	0.5032±0.0019 (2)	* (10)	0.4997±0.0070 (3)	0.4629±0.0021 (1)	0.4678±0.0058 (4)	3.29
FF	0.0769±0.0007 (11)	0.4385±0.0029 (8)	0.0665±0.0009 (11)	0.0943±0.0009 (6)	0.0196±0.0007 (11)	0.0172±0.0007 (10)	0.2307±0.0060 (9)	9.43
SM	0.4692±0.0094 (6)	0.6421±0.0118 (2)	0.3479±0.0081 (7)	0.1637±0.0249 (4)	0.4926±0.0079 (4)	0.0664±0.0033 (9)	0.3684±0.0134 (6)	5.43
TabPC	0.4995±0.0073 (3)	0.5731±0.0062 (4)	0.4490±0.0040 (4)	0.4514±0.0019 (1)	0.5047±0.0079 (2)	0.3475±0.0023 (6)	0.5003±0.0061 (2)	3.14

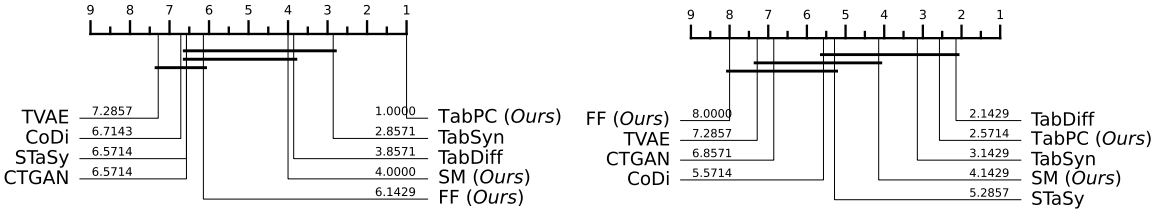


Figure 18: We present the corresponding CDDs for the "naive" implementation of α-PRECISION and β-RECALL for backwards compatibility with prior works. **TABPC is statistically higher performing than all other methods for α-PRECISION (naive; left)**, and falls in the top clique for β-RECALL (naive; right).