TabTreeFormer: Tabular Data Generation Using Hybrid Tree-Transformer

Jiayu Li[*†]
University of Illinois, Urbana-Champaign

Bingyin Zhao[†]
Pixocial Technology

Zilong Zhao
Betterdata AI

Uzair Javaid
Betterdata AI

Kevin Yee
Betterdata AI

Biplab Sikdar
National University of Singapore

Abstract

Transformers have shown impressive results in tabular data generation. However, they lack domain-specific inductive biases which are critical for preserving the intrinsic characteristics of tabular data. They also suffer from poor scalability and efficiency due to quadratic computational complexity. In this paper, we propose TabTreeFormer, a hybrid transformer architecture that integrates inductive biases of tree-based models (e.g., non-smoothness and non-rotational invariance) to effectively handle the discrete and weakly correlated features in tabular datasets. To improve numerical fidelity and capture multimodal distributions, we introduce a novel tokenizer that learns token sequences based on the complexity of tabular values. This reduces vocabulary size and sequence length, yielding more compact and efficient representations without sacrificing performance. We evaluate TabTreeFormer on nine diverse datasets, benchmarking against eight generative models. We show that TabTreeFormer consistently outperforms baselines in utility, fidelity, and privacy metrics with competitive efficiency. Notably, in scenarios prioritizing data utility over privacy and efficiency, the best variant of TabTreeFormer delivers a 44% performance gain relative to its baseline variant. Our code is available at: https://github.com/li-jiayu-ljy/tabtreeformer.

1 Introduction

Tabular data is a prevalent data modality in real-world applications (e.g., healthcare (Dash et al., 2019), financial services (Assefa et al., 2021), etc.), yet are heavily underexploited due to privacy concerns (EU, 2016). Fortunately, synthetic data offers an alternative to the utilization of tabular data by modeling the characteristics of real data and reducing the risk of data breach (PDPC, 2024), thus drawing considerable attention in recent years.

State-of-the-art (SOTA) research shows that transformers such as autoregressive transformers (Borisov et al., 2023), masked transformers (Gulati and Roysdon, 2023), and diffusion models with transformers (Zhang et al., 2024) have achieved impressive performance in tabular data generation, allowing synthetic data to empower a variety of fields (Hernandez et al., 2022; Assefa et al., 2021). However, unlike the application of transformers in other research areas such as computer vision (Dosovitskiy et al., 2021) and natural language processing (Vaswani et al., 2017), existing transformer models for synthetic tabular data often overlook domain-specific priors (i.e., inductive biases). For example, CvT (Wu et al., 2021) introduces convolutional embedding and projection to transformers to boost vision performances, and Transformer-XL (Dai et al., 2019) introduces segment-level recurrence to transformers to capture longer-term dependencies. While vision and language models have enjoyed the performance boost introduced by domain-specific priors, less exploration has been conducted to leverage them in tabular generative transformers. Moreover, transformers suffer from poor scalability due to quadratic computational complexity. This raises two interesting questions:

1. What inductive biases are beneficial to the quality of synthetically generated tabular data?
2. How can one exploit these inductive biases to improve the generative transformer?

To answer the questions, we propose TabTreeFormer, a hybrid transformer incorporating a tree-based model and a

[*]Corresponding author: jiayul11@illinois.edu.
[†]Work done while working at Betterdata AI and National University of Singapore.
Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

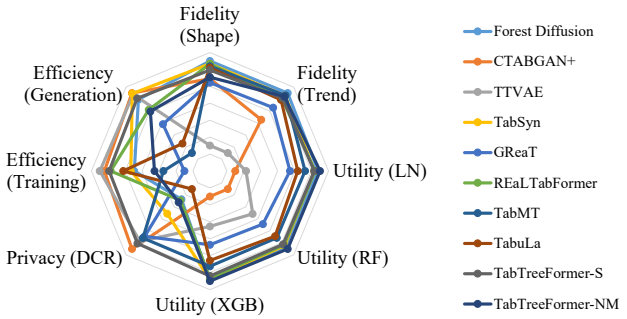

Figure 1: Performance comparison of TabTreeFormer (Ours) with SOTA tabular generative models in utility, fidelity, privacy, and efficiency metrics. TabTreeFormer-S achieves the best balance as the only big near-regular octagon, and TabTreeFormer-NM achieves the best utility.

novel tokenizer to handle tabular-specific inductive biases. We also leverage the limited per-dimension semantic meaning (i.e., each dimension corresponds to at most one feature) of typical tabular data to enable a significant reduction in model size with our tokenizer design.

Tree-based models excel at tabular classification and regression tasks (Shwartz-Ziv and Armon, 2022; Gorishniy et al., 2021), which is attributed to their inductive biases that can capture tabular characteristics such as *non-smoothness* and potentially *low-correlation* (Grinsztajn et al., 2022). Non-smoothness is due to the existence of discrete features and non-smooth relations between discrete and/or continuous features. Trees effectively model these by learning piecewise constant functions, unlike neural networks that typically learn smoother, low-frequency functions (Rahaman et al., 2019). Low-correlated features, often uninformative, contribute minimally to downstream tasks such as classification and regression. Due to the non-rotationally invariant nature of tree-based models (Ng, 2004), trees are more robust against low-correlated features, while neural networks are biased towards stronger correlations everywhere. Inspired by the success of tree models on tabular tasks, we propose to employ such priors to facilitate the performance on tabular generative tasks.

However, tree models do not inherently capture *multimodal distributions* in continuous features (i.e., features whose probability density functions have multiple modes or peaks). This limitation poses a significant challenge for tabular generative modeling. Inspired by (Xu et al., 2019) that addresses this issue via multimodal decomposition, we propose dual-quantization tokenization for transformers. The first quantization uses K-Means clustering (MacQueen, 1967) to model multimodal distributions, while the second employs a separate quantile-based quantization to achieve more precise representation of numerical values. Unlike typical word tokens, quantized tokens possess ordinal relationships. To effectively capture these relationships, we design and implement custom embeddings and loss functions.

Our contributions are summarized as follows:

- To the best of our knowledge, we are the first to introduce tabular-specific inductive biases to transformers with a tree-based model for improving table generation.
- We propose a dual-quantization tokenizer with ordinal-aware embeddings and loss functions, enabling transformers to model multimodal continuous features and generate high-quality synthetic data.
- Compared to 8 baselines across 9 datasets, TabTreeFormer achieves an outstanding balance in quality, privacy, and efficiency, as shown in Fig. 1. It also achieves up to 54% and 44% gain over the best deep and general baselines respectively, when prioritizing utility over other metrics.

2 Related Works

2.1 Tabular Data Generation

Early works on tabular data generation use MLPs and CNNs as backbone architectures with GAN and VAE as generation methods (Park et al., 2018; Xu et al., 2019; Zhao et al., 2021, 2024). Recent works now use transformers with auto-regression, masked modeling, and diffusion as generative paradigms (Borisov et al., 2023; Zhang et al., 2024; Gulati and Roysdon, 2023). For example, TabMT (Gulati and Roysdon, 2023) employs a masked transformer with ordered embedding and achieves good utility and scalability in downstream tasks. TabSyn (Zhang et al., 2024) encodes tabular data into a latent space and generates tabular data using a diffusion model, demonstrating impressive fidelity and utility. Despite their enhanced performance, they overlook inductive biases to capture non-smoothness and low-correlated features that are crucial to preserving the intrinsic characteristics of tabular data. Moreover, Xu et al. (2019) highlight that capturing multimodal distribution in continuous features is a critical challenge for tabular generative models. CTGAN (Xu et al., 2019) uses variational Gaussian mixture model (Bishop, 2006) to decompose multimodal values, while TabDiff (Shi et al., 2025) introduces a multimodal stochastic sampler. In this paper, we propose a tokenizer that models the multimodal distribution and injects this prior into an auto-regressive transformer for improved tabular data generation.

2.2 Tree-based Models for Tabular Data

Tree-based models like XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018) dominate tabular

tasks due to their strong inductive biases, enabling high predictive performance, efficiency, and scalability. While transformer-based models (Huang et al., 2020; Arik and Pfister, 2021; Hollmann et al., 2023) show promise, they often underperform due to lack of such biases (Shwartz-Ziv and Armon, 2022; Gorishniy et al., 2021). Tree-based generative models (Watson et al., 2023; McCarter, 2024; Jolicoeur-Martineau et al., 2024), though efficient, struggle with synthetic data quality or suffer from privacy leakage risks. In this paper, we bridge the gap by combining tree-based priors with transformers to inject tabular-specific inductive bias and enhance tabular generative performance.

2.3 Tabular Tokenizers

Transformers require a tokenizer to convert tabular data into suitable input tokens. Some methods map data to a continuous space by redesigning embedding layers (Arik and Pfister, 2021; Gorishniy et al., 2022; Zhang et al., 2024), while others tokenize into discrete sequences with minimal embedding changes (Borisov et al., 2023; Solatorio and Dupriez, 2023; Gulati and Roysdon, 2023; Zhao et al., 2023), which align better with NLP practices. In this paper, we adopt the latter to optimize and simplify tabular tokenization. This approach converts continuous values to discrete tokens, which have ordinal relations, such that the greatness of the token IDs can be compared. To capture this relation, two aspects where a model can be modified to cater for the ordinal token space are: i) embedding (input, also output for causal language models) and ii) loss (output).

Ordinal Embedding. One of the most well-known continuous embedding methods is the positional embedding based on trigonometric functions (Vaswani et al., 2017) that has been adopted for ordinal tokens (Narvekar and Mehta, 2024). However, their periodic nature is better suited for language positions than general ordinal data. Other methods often rely on non-ordinal embeddings (Li et al., 2021; Gulati and Roysdon, 2023) or require high-precision raw values (Gorishniy et al., 2022; Gulati and Roysdon, 2023). We propose a function-based ordinal embedding that sidesteps both demands.

Ordinal Loss. Prior work on ordinal regression often targets a small number of classes (Niu et al., 2016; Hou et al., 2016; Cao et al., 2020; Shi et al., 2023), or modifies cross-entropy loss to account for token distances (Diaz and Marathe, 2019; Nachmani et al., 2025; Castagnos et al., 2022; Zhang et al., 2023). Building on core ideas from prior work, we propose a custom loss tailored to our setting.

3 TabTreeFormer

Our goal is to improve generative modeling of tabular data by introducing domain-specific inductive biases. Let $\mathbf{X}$ be the training dataset with n rows, m_d discrete, and m_c continuous features. To generate synthetic tabular data $\mathbf{X}'$ similar to $\mathbf{X}$, we propose TabTreeFormer, a model composed of three components: a tree-based model, a tokenizer, and a transformer, as shown in Fig. 2. The tree model encodes tabular-specific inductive biases; the tokenizer captures multimodal distributions while reducing vocabulary and sequence length; and the transformer learns priors from both to generate high-quality synthetic data. More details on training and generation are provided in Appendix.

3.1 Tree-based Model

To inject tabular-specific inductive biases, we incorporate a tree-based model into the transformer ("Tree-based Model" in Fig. 2) by augmenting each row's token sequence with leaf indices from a fitted tree model $\mathcal{T}$ with T trees. Let $l_k \in \mathbb{N}$ be the number of leaves in k-th tree, where $k \in \{1, \ldots, T\}$. In tree $\mathcal{T}$, the leaf index matrix $\mathbf{J} = [j_{ik}] \in \mathbb{N}^{n \times T}$ captures the position of each row $\mathbf{X}_i$ in each tree. These indices encode non-smooth and non-rotationally invariant structure, enabling the model to possess these inductive biases of tabular data. By prepending leaf indices to input tokens, we transfer tree-based inductive biases to the transformer. During inference, these indices also act as prompts and guide towards more realistic data generation.

Many early tabular generation frameworks use conditional generation (Xu et al., 2019; Zhao et al., 2021), which improves the generation quality noticeably. These conditions are based on the values of a specific column in the dataset. In TabTreeFormer, we extend the condition of generation to clusters of the data and allow multiple concurrent conditions in place. Conceptually, each tree of a well-trained tree-based model provides an informative clustering of the training set, and the leaf index matrix provides a number of such clusterings. Therefore, providing the leaf indices as prompts, serving as conditions for generation, where multiple concurrent conditions (clusters) are provided and each condition features the clustering on one tree, is expected to be an effective generation performance booster.

3.2 Tokenizer

To model multimodal distributions and reduce both vocabulary size and sequence length, we design a tailored dual-quantization tokenizer for tabular data ("Tokenizer" in Fig. 2). For each continuous feature, we quantize the values so that it is of a more natural data format for transformers to learn. However, unlike TabMT (Gulati and Roysdon, 2023) that uses a single quantizer, we use a dual-quantizer (i.e.,

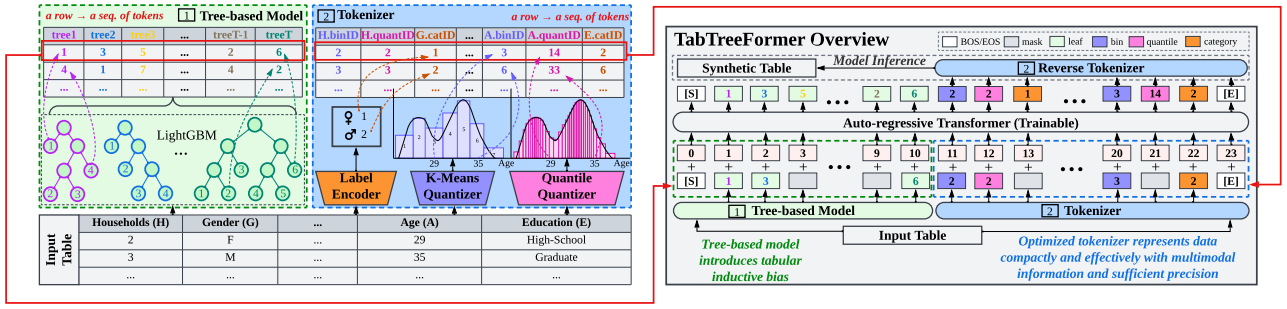

Figure 2: Overview of TabTreeFormer (data flow: left → right, bottom → top). It consists of 3 components: i) **a tree-based model** that introduces tabular-specific inductive biases; ii) **a tokenizer** that efficiently and compactly represents data while capturing multimodal distributions; iii) **a transformer model** that learns the priors extracted from the tree and tokenizer to generate synthetic data.

Table 1: Illustration of tokens in TabTreeFormer. "Size" indicates the number of tokens of this type. Recall Fig. 2 for the their colors and usage. "N" in "Format" can be any index in the range; e.g., leaf 3 ($3 < n_l$) is represented as [leaf3], corresponding to the light green tokens in Fig. 2 with value 3.

Type (Alias)	Size	Format	Source	Description
Leaf	$n_l = \max_{i\in\{1,2,\ldots,T\}} l_i$	[leafN]	Tree-based model $\mathcal{T}$	Leaf index in a tree
Categorical (cat)	$n_c = \max_{i\in\{1,2,\ldots,n\}} c_i$	[catN]	Data tokenizer $\mathcal{D}$ (categorical columns)	Category ID
Cluster (bin)	$n_b = \max_{i\in\{1,2,\ldots,n\}} b_i$	[binN]	Data tokenizer $\mathcal{D}$ (numeric columns)	Bin ID in K-Means quantizer
Quantile (quant)	$n_q = \max_{i\in\{1,2,\ldots,n\}} q_i$	[quantN]	Data tokenizer $\mathcal{D}$ (numeric columns)	Quant. ID in quant. quantizer
Special	3	[BOS] [EOS] [mask]	Classical special tokens for LMs	BOS, EOS, and mask tokens

double quantizers) to capture multimodal continuous distribution and describe the precise values respectively. Transformers do not expect near-Gaussian-distributed input tokens, so we capture the multimodal distribution simply by a K-Means quantizer (MacQueen, 1967) with a small number of clusters (e.g., $K = 10$). Following that, the original value is then quantized into a large number of quantiles (e.g., $Q = 1000$), based on which original values are recovered, and thus the values are highly precise. Employing the dual-quantization tokenization scheme, each numeric value is encoded into two discrete values: a bin ID in the K-Means quantizer and a quantile ID in the quantile quantizer, allowing two tokens to represent a numeric value with valuable and sufficient information. Categorical features are simply tokenized by label encoding, requiring 1 token. Formally, a reversible data tokenizer $\mathcal{D}$ can be fitted on $\mathbf{X}$. Let $c_i \in \mathbb{N}, b_i \in \{0, \ldots, K\}, q_i \in \{0, \ldots, Q\}$ denote the number of categories, K-Means bins, and quantiles in i-th column respectively, where $c_i = 0$ in continuous features and $b_i = q_i = 0$ in categorical features. Then, the domain for a categorical value is $\{1, \ldots, c_i\} \subseteq \mathbb{N}^+$, and the domain for a continuous value is $\{1, \ldots, b_i\} \times \{1, \ldots, q_i\} \subseteq (\mathbb{N}^+)^2$. A summary is seen in Table 1.

3.3 Transformer

TabTreeFormer is simply trained as an auto-regressive transformer. The input token sequence is constructed by concatenating the outcome from the tree model and tokenizer. It requires 5 types of tokens and a vocabulary size of $V = n_l + n_c + n_b + n_q + 3$. The number of tokens in a sequence corresponding to one row is $L = 2 + T + m_d + 2m_c$, including BOS and EOS tokens. Both the vocabulary size and sequence length are much smaller than the need for natural language models. A shared set of leaf tokens across all trees and similarly shared sets of category, bin, and quantile tokens across all features are used. Distinctions between different trees and features are effectively encoded through positional information, leveraging the positional embeddings inherent in most transformers.

A key observation on the construction of token sequence is that the valid range of tokens at each position is fixed for a given table. Consequently, tokens can be sampled exclusively from this precomputed set of valid options during generation. This ensures that every generated token sequence corresponds to a valid row, eliminating the need for rejecting invalid sequences (Borisov et al., 2023; Solatorio and Dupriez, 2023; Zhao et al., 2023), thus significantly accelerating the sampling process.

Additionally, transformers are prone to memorization, particularly when trained on tabular data, where datasets are much smaller compared to typical natural language corpora (Borisov et al., 2023). To avoid memorization, we heavily mask the input and maintain the target output un-

masked. Moreover, we evenly split the training set into 2 subsets, one neural network (i.e., transformer in TabTreeFormer) is trained on one subset and validated on the other. Consequently, the validation loss can be a criterion for early stopping to avoid overfitting, too. When generating samples, we sample from the two networks with equal probability.

Theorem 1. *Given a real dataset* $\mathbf{X}$*, train* M *generative models* $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_M)$ *on an* M*-partition of it* $(\mathbf{X} = [\mathbf{X}^{[1]}; \mathbf{X}^{[2]}; \ldots; \mathbf{X}^{[M]}])$*. If the generators are well-trained (distribution* p *of the corresponding partition is learned), then to sample* $\mathbf{X}'$*, sample data from* $\mathcal{G}_i$ *for a probability of* $\left|\mathbf{X}^{[i]}\right| / |\mathbf{X}|$ *where* $|\cdot|$ *means the number of rows, we will have the resulting* $p(\mathbf{X}') = p(\mathbf{X})$*.*

Theorem 1 validates a generalized case of the method with two evenly split subsets described above, and can be trivially proven by the chain rule, so the proof is omitted. The rest of the training and generation settings are similar to the training and generation of classical causal language models, except for a modified embedding and loss function to account for the ordinal relations between quantile tokens, with details introduced in next Section.

3.4 Embedding and Loss Function for Ordinal Tokens

Quantile Embeddings (QE) with Ordinal Relations. The known ordinal relation between quantile tokens resembles the known relative sequential relation between positions. Therefore, analogous to function-generated positional embeddings (Vaswani et al., 2017), we use function-generated embeddings for quantile tokens. Unlike positional tokens, which emphasize relative values, quantile tokens focus on absolute values, so we replace periodic trigonometric functions with non-linear monotonic scaled sigmoid functions for embedding generation. The embedding value generators are provided in Equation 3, where $i \in \{0, \ldots, Q-1\}$ stands for the quantile ID, and $d \in \{0, \ldots, D-1\}$ stands for the embedding dimensions. More detailed intuition and description of the function are provided in Appendix.

$$S_d = \left\lfloor \frac{1 + \sqrt{1 + 4d}}{2} \right\rfloor \in \mathbb{N}^+ \qquad \text{(scale factor)} \qquad (1)$$

$$O_d = \frac{-4S_d^3 + (4d+2)S_d}{2S_d - 1} \in [-2S_d, 2S_d] \quad \text{(offset)} \quad (2)$$

$$QE_{id} = \text{sigmoid}\left(4S_d\left(\frac{i}{Q} - \frac{1}{2}\right) + O_d\right) \qquad (3)$$

Theorem 2. *Following the quantile embedding values in Equation 3, and let the embedded vector of quantile* i *be* $\mathbf{q}_i = \begin{pmatrix} QE_{i0} & \cdots & QE_{i(D-1)} \end{pmatrix} \in (0,1)^D$*. For any* $p \geq 1$*, given quantile IDs* $i, j, k \in \{0, \ldots, Q-1\}$*, and* j, k *are on the same side of* i *(i.e.,* $(i-j)(i-k) \geq 0$*), then* $|i-j| < |i-k|$ *if and only if* $\|\mathbf{q}_i - \mathbf{q}_j\|_p < \|\mathbf{q}_i - \mathbf{q}_k\|_p$*, where* $\|\cdot\|_p$ *stands for the* p*-norm of a vector.*

Theorem 2 shows that the trend of the distance between the embedded vectors is consistent with the difference between quantile IDs. The mathematical proof is provided in Appendix. At different embedding dimensions, we apply different slopes (by scale factor) and intercepts (by offset) on the input to $\mathrm{sigmoid}$. Note that while the meaning of positions in a transformer is fixed, the interpretation of quantiles can vary across datasets. As a result, instead of fixing the embedding values as in Equation 3, we initialize the values by the equations, and they are updated during training. Non-quantile tokens will use a typical word token embedding layer of transformers.

Ordinal Cross-Entropy Loss. Cross-entropy loss (CEL) is typically used as the optimization objective of transformers. However, the quantized quantile tokens retain an inherent ordinal property, where closer IDs correspond to closer values. This relationship is not captured by standard CEL, where all classes as equally distinct. To address this, we replace the vanilla CEL with a specialized ordinal CEL for quantile tokens. Formally, let the predicted logits for a token be $\mathbf{z} \in \mathbb{R}^V$, the probability after softmax be $\mathbf{p} \in [0,1]^V$ ($\|\mathbf{p}\|_1 = 1$), and the target label be $t \in \{1, \ldots, V\}$. We define *ordinal cross-entropy loss* (OCEL) as a weighted version of CEL. These weights are applied to unnormalized probabilities in both the numerator and denominator, depending on the current- and target-class pairs. It contrasts with the classical weighted CEL, which weights the loss per class based solely on the target class and applies to the numerator only. Formally, we write it as Equation 5 (recall the classical CEL in Equation 4), where w_{ti} denotes the weight for z_i, where t is the target class.

$$\mathcal{L}_{\text{ce}}(\mathbf{z}, t) = -\log \frac{e^{z_t}}{\sum_{i=1}^{V} e^{z_i}} = -\log p_t \qquad (4)$$

$$\mathcal{L}_{\text{oce}}(\mathbf{z}, t) = -\log \frac{w_{tt} e^{z_t}}{\sum_{i=1}^{V} w_{ti} e^{z_i}} \qquad (5)$$

Lemma 1. *Given* $\mathbf{w} > \mathbf{0}$ *independent of* $\mathbf{z}$*, OCEL is optimized when* $z_t \to \infty$ *and* $z_i \to -\infty, \forall i \neq t$*.*

Theorem 3. *Let* $f(|t-i|)$ *be a variant of the distance between current class* i *and target class* t *when* f *is non-negative and monotonically increasing. Let the weighted sum of this variant of distances to target incurred by some logit* $\mathbf{z}$ *be* $D(\mathbf{z}) = \sum_{i=1}^{V} p_i w_i$*. If* $w_{ti} = f(|t-i|)$ *for some* f*, then for* $\mathbf{z}, \mathbf{z}^*$ *with* $\mathcal{L}_{ce}(\mathbf{z}, t) = \mathcal{L}_{ce}(\mathbf{z}^*, t)$ *and* $D(\mathbf{z}) < D(\mathbf{z}^*)$*, we must have* $\mathcal{L}_{oce}(\mathbf{z}, t) < \mathcal{L}_{oce}(\mathbf{z}^*, t)$*.*

Both Lemma 1 and Theorem 3 are intuitive, and we provide rigorous proofs in Appendix. Theorem 3 implies that the

Table 2: Averaged MLE performance of different models on all 9 datasets. RE stands for relative error (w.r.t. score on real data). Avg. stands for the average raw MLE score. The best scores and the second best scores are highlighted in bold with and without underscore, respectively.

	ML	Real	FD	CTAB+	TTVAE	TabSyn	GReaT	RTF	TabMT	TabuLa*	TTF-S	TTF-L	TTF-NM
RE† (↓)	all		**0.020**$_{\pm 0.033}$	0.253$_{\pm 0.287}$	0.203$_{\pm 0.335}$	0.024$_{\pm 0.030}$	0.117$_{\pm 0.178}$	0.027$_{\pm 0.042}$	0.062$_{\pm 0.103}$	0.041$_{\pm 0.051}$	0.031$_{\pm 0.040}$	0.021$_{\pm 0.035}$	**0.011**$_{\pm 0.021}$
	LN		**0.014**$_{\pm 0.027}$	0.231$_{\pm 0.260}$	0.249$_{\pm 0.476}$	0.025$_{\pm 0.036}$	0.095$_{\pm 0.131}$	0.023$_{\pm 0.033}$	0.056$_{\pm 0.098}$	0.037$_{\pm 0.065}$	0.025$_{\pm 0.035}$	0.019$_{\pm 0.032}$	**0.007**$_{\pm 0.015}$
	RF		0.020$_{\pm 0.027}$	0.256$_{\pm 0.293}$	0.184$_{\pm 0.275}$	0.019$_{\pm 0.024}$	0.127$_{\pm 0.202}$	0.025$_{\pm 0.037}$	0.064$_{\pm 0.107}$	0.045$_{\pm 0.049}$	0.031$_{\pm 0.037}$	**0.018**$_{\pm 0.045}$	**0.013**$_{\pm 0.027}$
	XGB		0.026$_{\pm 0.044}$	0.272$_{\pm 0.337}$	0.175$_{\pm 0.242}$	0.028$_{\pm 0.031}$	0.130$_{\pm 0.210}$	0.034$_{\pm 0.056}$	0.067$_{\pm 0.116}$	0.041$_{\pm 0.046}$	0.036$_{\pm 0.050}$	**0.026**$_{\pm 0.032}$	**0.014**$_{\pm 0.020}$
Avg. (↑)	all	0.899$_{\pm 0.104}$	**0.884**$_{\pm 0.122}$	0.685$_{\pm 0.285}$	0.747$_{\pm 0.312}$	0.880$_{\pm 0.119}$	0.807$_{\pm 0.215}$	0.878$_{\pm 0.129}$	0.853$_{\pm 0.168}$	0.840$_{\pm 0.129}$	0.875$_{\pm 0.126}$	0.881$_{\pm 0.113}$	**0.889**$_{\pm 0.106}$
	LN	0.891$_{\pm 0.125}$	**0.881**$_{\pm 0.140}$	0.695$_{\pm 0.272}$	0.719$_{\pm 0.405}$	0.872$_{\pm 0.145}$	0.818$_{\pm 0.200}$	0.874$_{\pm 0.143}$	0.851$_{\pm 0.182}$	0.835$_{\pm 0.163}$	0.872$_{\pm 0.141}$	0.875$_{\pm 0.135}$	**0.885**$_{\pm 0.126}$
	RF	0.901$_{\pm 0.104}$	**0.885**$_{\pm 0.119}$	0.686$_{\pm 0.289}$	0.757$_{\pm 0.286}$	0.884$_{\pm 0.109}$	0.801$_{\pm 0.233}$	0.881$_{\pm 0.127}$	0.852$_{\pm 0.171}$	0.839$_{\pm 0.124}$	0.875$_{\pm 0.125}$	0.883$_{\pm 0.105}$	**0.888**$_{\pm 0.100}$
	XGB	0.906$_{\pm 0.094}$	**0.886**$_{\pm 0.122}$	0.673$_{\pm 0.326}$	0.765$_{\pm 0.262}$	0.883$_{\pm 0.113}$	0.803$_{\pm 0.236}$	0.880$_{\pm 0.131}$	0.854$_{\pm 0.171}$	0.848$_{\pm 0.116}$	0.878$_{\pm 0.127}$	0.885$_{\pm 0.112}$	**0.895**$_{\pm 0.101}$

Table 3: Averaged fidelity performance of different models on all 9 datasets. The best scores and the second best scores are highlighted in bold with and without underscore, respectively.

	FD	CTAB+	TTVAE	TabSyn	GReaT	RTF	TabMT	TabuLa	TTF-S	TTF-L	TTF-NM
Shape	**0.931**$_{\pm 0.047}$	0.890$_{\pm 0.067}$	0.736$_{\pm 0.150}$	**0.925**$_{\pm 0.052}$	0.881$_{\pm 0.062}$	0.924$_{\pm 0.067}$	0.919$_{\pm 0.052}$	0.916$_{\pm 0.076}$	0.910$_{\pm 0.037}$	0.915$_{\pm 0.042}$	0.894$_{\pm 0.081}$
Trend	**0.922**$_{\pm 0.064}$	0.813$_{\pm 0.122}$	0.678$_{\pm 0.219}$	0.911$_{\pm 0.084}$	0.862$_{\pm 0.088}$	0.899$_{\pm 0.091}$	0.912$_{\pm 0.065}$	0.894$_{\pm 0.085}$	0.901$_{\pm 0.069}$	**0.913**$_{\pm 0.076}$	0.908$_{\pm 0.098}$

OCEL penalizes (value being smaller) classes closer to the target class less than farther ones. Any definition of $w_{ti} = f(|t-i|)$ satisfying the monotonicity constraint works for OCEL technically. In this paper, we define the weight w_{ti} for z_i to be obtained by Equation 6.

$$w_{ti} = 1 + m - e^{-\frac{(t-i)^2}{(V\sigma)^2}} \tag{6}$$

where $\sigma = 0.005$ is a scaling factor of the distance, and $m = 0.5$ is the minimum weight (used on $i = t$). Detailed explanation and analysis of it can be found in Appendix.

Note that not all tokens are part of the ordinal relation, so the OCEL is applied only to valid quantile tokens at the corresponding positions. To enable the model to learn to generate valid quantile tokens, we introduce an additional modified CEL to distinguish valid tokens from invalid ones, which complements OCEL. See Appendix for the exact overall loss formula.

4 Experiments

4.1 Experimental Setup

Basic Setup. For all datasets, we split train and test sets in $4:1$, and repeat each experiment 3 times and report the results. We conduct all experiments on a machine equipped with 1 NVIDIA RTX 4090 and 20 CPU cores. Detailed environment setting can be found in Appendix.

TabTreeFormer Configuration. We use LightGBM (Ke et al., 2017) whose hyperparameters are tuned by Optuna (Akiba et al., 2019) as the tree-based model. We exploit Distill-GPT2 (Sanh et al., 2019) as the transformer backbone, following the practice in prior works (Borisov et al., 2023; Solatorio and Dupriez, 2023). TabTreeFormer (TTF) is experimented with two major configurations: S (small) and L (large), with the number of trainable parameters of approximately 5M and 40M, respectively. When privacy is not crucial concerns, we remove the masks of TTF-L, which we call TTF-NM (**n**o **m**ask). In ablation study, we use TabTreeFormer-S. More implementation details can be found in Appendix.

Datasets. Experiments are conducted on 9 datasets with diverse sizes and characteristics from OpenML (Vanschoren et al., 2013): adult, bank, boston, breast, credit, diabetes, iris, qsar, and wdbc. Only credit and diabetes are used for ablation study. Details of datasets are summarized in Appendix.

Baseline Models. We compare the performance of TabTreeFormer against 8 SOTA methods in tabular data generation, including non-neural networks, GANs, VAEs, diffusion models, and auto-regressive transformers (ART) or masked transformers: Forest Diffusion (FD) (Jolicoeur-Martineau et al., 2024), CTAB-GAN+ (CTAB+) (Zhao et al., 2024), TTVAE (Wang and Nguyen, 2025), TabSyn (Zhang et al., 2024), GReaT (Borisov et al., 2023), REaLTabFormer (RTF) (Solatorio and Dupriez, 2023), TabMT (Gulati and Roysdon, 2023), and TabuLa (Zhao et al., 2023). Justification of the baseline choices and implementation details are described in Appendix.

Metrics. We evaluate TabTreeFormer using utility, fidelity, privacy, and efficiency, which are standard metrics in tabular data generation (Xu et al., 2019; Zhao et al., 2024; Borisov et al., 2023; Zhang et al., 2024; Shi et al., 2025). Detailed implementation of metrics can be found in Appendix.

- **Utility** exhibits the quality of synthetic data by testing its performance on downstream tasks. We evaluate based on the established Train-on-Synthetic, Test-on-

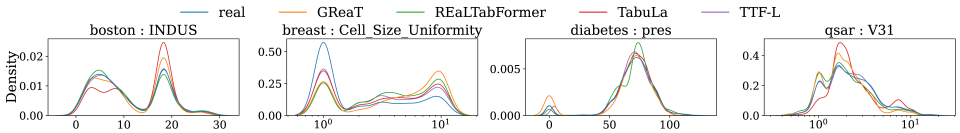

Figure 3: Marginal densities of representative multimodal continuous columns from baseline ART and TTF. All have a Distill-GPT2 backbone.

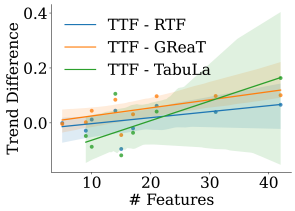

Figure 5: Feature count vs. Trend score improvement of TTF from ART baselines.

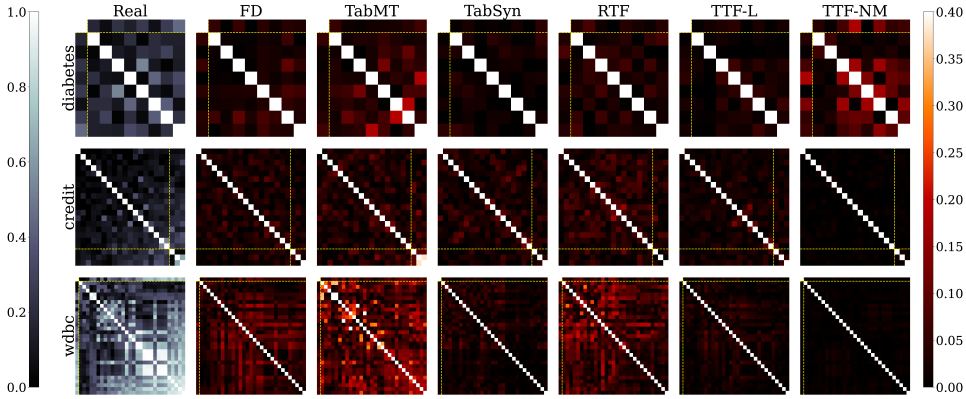

Figure 4: TTF versus top-4 baselines in pair-wise correlation. Real (absolute) correlation values are presented on the left, and the absolute error in correlation values in synthetic data from different models, capped at 0.4 for visibility, are shown at the right (the darker the better). The left-top features (divided by yellow line) are categorical and the right-bottom are numeric.

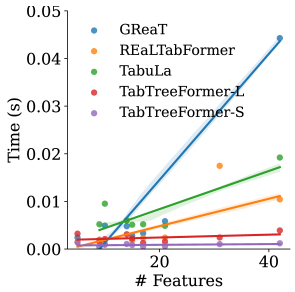

Figure 6: Number of features vs. average generation time per (valid) row of TTF and ART baselines.

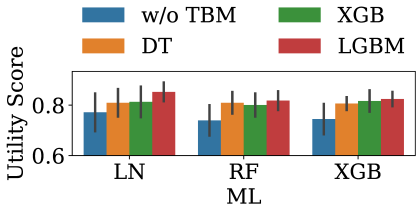

Figure 7: Utility of different tree-based model in TTF. The x-axis shows the downstream model type.

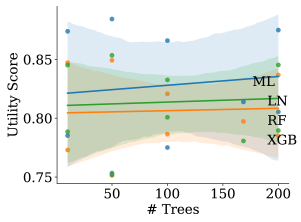

Figure 8: Utility of different number of trees in the tree-based model in TTF.

Table 4: Design component ablation. Average utility scores are reported. All experiments are based on variants of TabTreeFormer-S.

ML	all	LN	RF	XGB
TTF-S	0.832	0.853	0.818	0.824
gen. w/o tree	0.792	0.808	0.780	0.789
w/o OCEL	0.809	0.825	0.802	0.800
w/o QE	0.801	0.820	0.783	0.799

Real (TSTR) machine learning efficacy (MLE) evaluation framework (Xu et al., 2019), namely, synthetic data generated based on real training data is tested on a hold-out real test set. Three models are used: logistic regression (e.g., linear–LN), random forest (RF), and XGBoost (XGB). Classification performance is evaluated by weighted AUC and regression is evaluated by R^2. For both metrics, higher scores indicate better performance. MLE is the most widely used synthetic tabular data generation evaluation metric, which is reported in all baselines, so we treat this as the core synthetic data quality score.

- **Fidelity** shows the cosmetic discrepancy between the real and synthetic data. It is evaluated via "Shape" and "Trend" metrics (sdm, 2023; Shi et al., 2025). "Shape" measures the similarity of marginal distribution density for each column, and "Trend" measures the fidelity in correlation between column pairs. Higher "Shape" and "Trend" values indicate better data fidelity.
- **Privacy** is a crucial criterion when the synthetic data serves the use case of privacy-preserving data sharing. We use the distance to the closest record (DCR) (Zhao et al., 2021) to evaluate the privacy of synthetic data. We compare the DCRs from synthetic data and from hold-out real (test) data to the real training data. Mann-Whitney U Test (Mann and Whitney, 1947) is applied on these two sets of DCRs and the null hypothesis assumes the former DCRs are no smaller than the latter. Privacy is at risk of disclosure when p-values are smaller than 0.05.
- **Efficiency** showcases the model sizes and computation

Table 5: Privacy performances. The first two rows are the average and minimum p-values, and the last row shows the number of violations (i.e., $p < 0.05$). The closer the p-value to 0, the higher the risk of privacy leakage. Non-zero violations are highlighted in red, indicating privacy risks.

	FD	CTAB+	TTVAE	TabSyn	GReaT	RTF	TabMT	TabuLa	TTF-S	TTF-L	TTF-NM
Avg. p	0.380	1.000	0.904	0.559	0.843	0.394	0.865	0.251	0.933	0.745	0.417
Min. p	0.000	1.000	0.382	0.029	0.082	0.000	0.259	0.000	0.662	0.214	0.000
# vio.	5	0	0	1	0	3	0	5	0	0	3

time for training and generation of different models. Under comparable performance, smaller models and faster computation are favored.

4.2 Result Analysis

Utility: Table 2 shows TabTreeFormer-NM consistently outperforms all baselines on MLE, demonstrating its superior synthetic-data generation. The margin grows with stronger downstream models (XGB $>$ RF $>$ LN), underscoring TabTreeFormer 's ability to capture complex relations via tree-based modeling. Results on individual datasets are detailed in Table 10 in Appendix.

Fidelity Table 3 shows the fidelity results. TabTreeFormer achieves comparable performance to baseline models in "Shape" metric, especially to ART-based baseline models. Meanwhile, Fig. 3 shows that TabTreeFormer captures better (i.e., closer to real) multimodal distribution compared to other ARTs, which validates the effectiveness of our dual-quantization tokenizer design. TabTreeFormer-L outperforms all neural network baselines in "Trend", implying the superior capability of TabTreeFormer in learning inter-feature relations. We visualize the pair-wise correlation of some datasets in Fig. 4. While TabTreeFormer-NM is able to capture the correlations on certain datasets perfectly, its performance variance across datasets is large. In addition, TabTreeFormer tends to show a more significant improvement from ART baselines with more features, as referred in Fig. 5. This demonstrates the advantage of introducing the inductive bias to address low-correlations so that models learn better by filtering out less correlated features, which is common for dataset with more features.

Privacy: As summarized in Table 5, TabTreeFormer-S and L are privacy-resilient in all 9 datasets. In comparison, FD has privacy issues in 5 out of 9 datasets, which is consistent with our analysis in **Related Works** on its potential privacy issues. As for ART baseline methods, all but GReaT demonstrates privacy risk, particularly TabuLa, which bears severe privacy leakage concerns in 5 out of 7 successfully run datasets. This result also verifies that the masking and early stopping of TabTreeFormer's design successfully protects privacy. Raw p-values per experiment are presented in Table 12 in Appendix.

[*]For TabuLa, 2/9 datasets fail to generate reasonable data under the default setting.

[†]The reported utility improvement in abstract and introduction is computed from this row.

Table 6: Average computation time (s) of TabTreeFormer and ART baselines. The best values are highlighted in **bold with underline**, the second best is in **bold without underline**.

	GReaT	RTF	TabuLa	TTF-S	TTF-L
Train	2470.344	**359.950**	718.260	**316.743**	773.362
Generate	45.998	24.932	74.702	**7.846**	**19.718**

Efficiency: The efficiency of TabTreeFormer is mainly demonstrated by its ability to achieve comparable performance with a significantly smaller model size. Although TabTreeFormer-L is better than TabTreeFormer-S in general, demonstrating the usefulness of a larger model, TabTreeFormer-S achieves comparable utility with the best-performing ART baseline (REaLTabFormer, which has only marginally better MLE results than TabTreeFormer-S), while the baseline models have more than 80M parameters.

Efficiency: TabTreeFormer's computation time compared to all ART baselines is shown in Table 6. TabTreeFormer consistently achieves faster generation. Raw training and generation time can be found in Table 14 in Appendix.

4.3 Ablation Study

A core design of TabTreeFormer is its integration of a tree-based model (TBM). As shown in Fig. 7, having a tree-based model integrated is helpful. However, the exact tree-based model used does not seem to make a significant difference, though a subtle trend that multiple-tree models (XGBoost and LightGBM) are better than single-tree model (decision tree (DT)) can be observed. Nevertheless, observing the effect of the number of trees on multiple-tree models, as shown in Fig. 8, no obvious trend can be observed either, though there is still a general but very subtle trend that more trees result in better performance. we also evaluate the ablations of (1) tree-leaf tokens as prompt (2) OCEL and (3) quantile embedding (QE) to TabTreeFormer-S. The result is shown in Table 4. All the three components have a significant impact on improving the performance. The mask ratio and temperature influence on TabTreeFormer-NM is also evaluated. The result shows that the larger ratio is masked or the larger temperature, the better privacy and worse utility, this result is discussed at Appendix due to page limit.

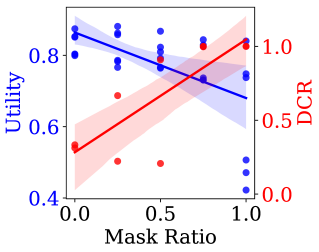

Figure 9: Quality & privacy trend of different mask ratios.

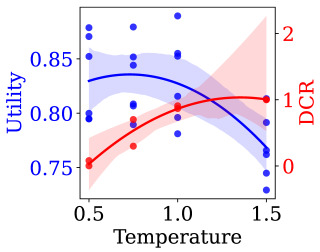

Figure 10: Quality & privacy of trend of different temperatures.

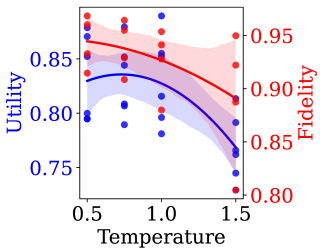

Figure 11: Utility & fidelity of trend of different temperatures.

4.4 TabTreeFormer-NM Settings.

Two core settings of NM different from L version is the mask ratio and temperature, so we also look at how the two parameters affect the performance, as shown in Fig. 9-11. The utility and DCR changes with mask ratio and temperature demonstrate clearly the dilemma between synthetic data quality and privacy, and as expected, the larger ratio is masked or the larger temperature, the better privacy and worse utility. Nevertheless, Fig. 11 shows a slight different trend on performance of utility and fidelity on temperature. As we maximize the synthetic data quality mainly by utility, it is likely that the temperatures is not maximized with the fidelity, which explains the reason for TabTreeFormer's score in fidelity (recall Table 3) not being the best.

5 Conclusion

In this paper, we propose TabTreeFormer, a hybrid tree-transformer based model for high-quality tabular data generation. It combines tree-based inductive biases with a dual-quantization tokenizer to model multimodal numeric distributions. Specialized loss and embeddings applied on quantized ordinal tokens. When evaluated on 9 datasets against 8 baselines, TabTreeFormer achieves stronger balance on quality, privacy, and efficiency, and particularly excels in data utility when privacy and efficiency are not prioritized.

References

(2023). *Synthetic Data Metrics*. DataCebo, Inc. Version 0.13.0.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., and Karypis, G., editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2623–2631. ACM.

Arik, S. O. and Pfister, T. (2021). TabNet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687.

Assefa, S. A., Dervovic, D., Mahfouz, M., Tillman, R. E., Reddy, P., and Veloso, M. (2021). Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20, New York, NY, USA. Association for Computing Machinery.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Borisov, V., Seßler, K., Leemann, T., Pawelczyk, M., and Kasneci, G. (2023). Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Cao, W., Mirjalili, V., and Raschka, S. (2020). Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognition Letters*, 140:325–331.

Castagnos, F., Mihelich, M., and Dognin, C. (2022). A simple log-based loss function for ordinal text classification. In Calzolari, N., Huang, C.-R., Kim, H., Pustejovsky, J., Wanner, L., Choi, K.-S., Ryu, P.-M., Chen, H.-H., Donatelli, L., Ji, H., Kurohashi, S., Paggio, P., Xue, N., Kim, S., Hahm, Y., He, Z., Lee, T. K., Santus, E., Bond, F., and Na, S.-H., editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4604–4609, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Krishnapuram, B., Shah, M., Smola, A. J., Aggarwal, C. C., Shen, D., and Rastogi, R., editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V.,

and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. In Korhonen, A., Traum, D. R., and Màrquez, L., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Dash, S., Shakyawar, S. K., Sharma, M., and Kaushik, S. (2019). Big data in healthcare: management, analysis and future prospects. *Journal of Big Data*, 6(1):54.

Diaz, R. and Marathe, A. (2019). Soft labels for ordinal regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

EU (2016). Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (General Data Protection Regulation). *OJ*, L 119.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Gorishniy, Y., Rubachev, I., and Babenko, A. (2022). On embeddings for numerical features in tabular deep learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24991–25004. Curran Associates, Inc.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18932–18943, Red Hook, NY, USA. Curran Associates, Inc.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 507–520, Red Hook, NY, USA. Curran Associates, Inc.

Gulati, M. and Roysdon, P. (2023). TabMT: Generating tabular data with masked transformers. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46245–46254, Red Hook, NY, USA. Curran Associates, Inc.

Harrison, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *JEEM*, 5(1):81–102.

Hernandez, M., Epelde, G., Alberdi, A., Cilla, R., and Rankin, D. (2022). Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

Hofmann, H. (1994). Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2023). TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hou, L., Yu, C., and Samaras, D. (2016). Squared earth mover's distance-based loss for training deep neural networks. *CoRR*, abs/1611.05916.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. S. (2020). TabTransformer: Tabular data modeling using contextual embeddings. *CoRR*, abs/2012.06678.

Jolicoeur-Martineau, A., Fatras, K., and Kachman, T. (2024). Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In Dasgupta, S., Mandt, S., and Li, Y., editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 1288–1296. PMLR.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, Red Hook, NY, USA. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kohavi, R. (1996). Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 202–207. AAAI Press.

Li, W., Huang, X., Lu, J., Feng, J., and Zhou, J. (2021). Learning probabilistic ordinal embeddings for uncertainty-aware regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13896–13905.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Le Cam, L. M. and Neyman, J., editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, Berkeley, CA.

Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50 – 60.

Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., and Consonni, V. (2013). Quantitative structure–activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information and Modeling*, 53(4):867–878. PMID: 23469921.

McCarter, C. (2024). Unmasking trees for tabular data. *CoRR*, abs/2407.05593.

Moro, S., Cortez, P., and Laureano, R. (2011). Using data mining for bank direct marketing: An application of the CRISP-DM methodology. In *Proceedings of the European Simulation and Modelling Conference*.

Nachmani, I., Genossar, B., Scharf, C., Shraga, R., and Gal, A. (2025). SLACE: A monotone and balance-sensitive loss function for ordinal regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(18):19598–19606.

Narvekar, A. and Mehta, S. (2024). Cat2Vec with position encoding: A new approach for handling ordinal features using learned embeddings with positional encoding. *International Journal of Computer Applications*, 186(44):9–15.

Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 78, New York, NY, USA. Association for Computing Machinery.

Niu, Z., Zhou, M., Wang, L., Gao, X., and Hua, G. (2016). Ordinal regression with multiple output CNN for age estimation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4920–4928.

Park, N., Mohammadi, M., Gorde, K., Jajodia, S., Park, H., and Kim, Y. (2018). Data synthesis based on generative adversarial networks. *Proc. VLDB Endow.*, 11(10):1071–1083.

PDPC (2024). Proposed guide on synthetic data generation. *Personal Data Protection Commission*. Accessed: 2024-12-26.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31, page 6639–6649, Red Hook, NY, USA. Curran Associates, Inc.

Qian, Z., Davis, R., and van der Schaar, M. (2023). Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 3173–3188, Red Hook, NY, USA. Curran Associates, Inc.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. C. (2019). On the spectral bias of neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

Shi, J., Xu, M., Hua, H., Zhang, H., Ermon, S., and Leskovec, J. (2025). TabDiff: a mixed-type diffusion model for tabular data generation. In *The Thirteenth International Conference on Learning Representations*.

Shi, X., Cao, W., and Raschka, S. (2023). Deep neural networks for rank-consistent ordinal regression based on conditional probabilities. *Pattern Anal. Appl.*, 26(3):941–955.

Shwartz-Ziv, R. and Armon, A. (2022). Tabular data: Deep learning is not all you need. *Inf. Fusion*, 81(C):84–90.

Smith, J., Everhart, J., Dickson, W., Knowler, W., and Johannes, R. (1988). Using the ADAP learning algorithm to forcast the onset of diabetes mellitus. *Proc. Annu. Symp. Comput. Appl. Med. Care*, 10.

Solatorio, A. V. and Dupriez, O. (2023). REaLTabFormer: Generating realistic relational and tabular data using transformers. *CoRR*, abs/2302.02041.

Street, W. N., Wolberg, W. H., and Mangasarian, O. L. (1993). Nuclear feature extraction for breast tumor diagnosis. In Acharya, R. S. and Goldgof, D. B., editors,

Biomedical Image Processing and Biomedical Visualization, volume 1905, pages 861 – 870. International Society for Optics and Photonics, SPIE.

Vanschoren, J., van Rijn, J. N., Bischl, B., and Torgo, L. (2013). OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, page 6000–6010, Red Hook, NY, USA. Curran Associates, Inc.

Wang, A. X. and Nguyen, B. P. (2025). TTVAE: Transformer-based generative modeling for tabular data generation. *Artificial Intelligence*, 340:104292.

Wang, Y., Feng, D., Dai, Y., Chen, Z., Huang, J., Ananiadou, S., Xie, Q., and Wang, H. (2024). HARMONIC: Harnessing llms for tabular data synthesis and privacy protection. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 100196–100212. Curran Associates, Inc.

Watson, D. S., Blesch, K., Kapar, J., and Wright, M. N. (2023). Adversarial random forests for density estimation and generative modeling. In Ruiz, F., Dy, J., and van de Meent, J.-W., editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 5357–5375. PMLR.

Wolberg, W. H. and Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *PNAS*, 87(23):9193–9196.

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., and Zhang, L. (2021). CvT: Introducing convolutions to vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22–31.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019). Modeling tabular data using conditional GAN. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32, pages 7335–7345, Red Hook, NY, USA. Curran Associates, Inc.

Zhang, H., Zhang, J., Shen, Z., Srinivasan, B., Qin, X., Faloutsos, C., Rangwala, H., and Karypis, G. (2024). Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zhang, S., Yang, L., Mi, M. B., Zheng, X., and Yao, A. (2023). Improving deep regression with ordinal entropy. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhao, Z., Birke, R., and Chen, L. Y. (2023). TabuLa: Harnessing language models for tabular data synthesis. *CoRR*, abs/2310.12746.

Zhao, Z., Kunar, A., Birke, R., and Chen, L. Y. (2021). CTAB-GAN: Effective table data synthesizing. In Balasubramanian, V. N. and Tsang, I., editors, *Proceedings of The 13th Asian Conference on Machine Learning*, volume 157 of *Proceedings of Machine Learning Research*, pages 97–112. PMLR.

Zhao, Z., Kunar, A., Birke, R., Van der Scheer, H., and Chen, L. Y. (2024). CTAB-GAN+: enhancing tabular data synthesis. *Frontiers in Big Data*, 6.

A Supplementary TabTreeFormer Model Description and Details

Section **End-to-end TabTreeFormer Pipeline** provides the full end-to-end training and generation process, and Section **Tree-based Model** and **Transformer** provide detailed model setup of core components.

A.1 End-to-end TabTreeFormer Pipeline

As a supplementary of Fig. 2 on the overall process of TabTreeFormer, and the textual description in Section **TabTreeFormer**, we provide the full end-to-end pseudocode in this section. Algorithm 1 and Algorithm 2 describe the end-to-end training and generation process of TabTreeFormer respectively.

Algorithm 1 Training of TabTreeFormer

Input: Tabular dataset $\mathbf{X}$ (n rows, m_d discrete features, m_c continuous features), s_0 shared initial steps, s total training steps
Output: Fitted tree-based model $\mathcal{T}$ and the *leaf index matrices* $\mathbf{J}_1, \mathbf{J}_2$, data tokenizer $\mathcal{D}$, and transformers $\mathcal{G}_1, \mathcal{G}_2$
$\mathcal{T} \leftarrow$ TuneAndFitTreeBasedModel($\mathbf{X}$) ▶ Let $T = \mathcal{T}$.n_trees, $n_l = \mathcal{T}$.max_n_leaves
$\mathbf{J} \leftarrow \mathcal{T}$.GetLeafIndex($\mathbf{X}$) ▶ Obtain leaf index matrix $\mathbf{J} \in \mathbb{N}^{n \times T}$
$\mathcal{D} \leftarrow$ FitEncoder($\mathbf{X}$) ▶ $n_c = \mathcal{D}$.max_n_cat, $n_b = \mathcal{D}$.max_n_bin, $n_q = \mathcal{D}$.max_n_quant
$\widetilde{\mathbf{X}} \leftarrow \mathcal{D}$.Encode($\mathbf{X}$) ▶ Get cat/bin/quant IDs $\widetilde{\mathbf{X}} \in \mathbb{N}^{n \times (m_d + 2m_c)}$
$\mathbf{Z} \leftarrow$ ToTokenIDs($[\mathbf{J}; \widetilde{\mathbf{X}}]$) ▶ Convert these IDs to token IDs, add BOS, EOS
$\mathcal{G} \leftarrow$ InitializeLM(n_pos $= 2 + T + m_d + 2m_c$, n_vocab $= 3 + n_l + n_c + n_b + n_q$) ▶ Also initialize embedding layer on quantile tokens with ordinal relations
for $i = 1, \ldots, s_0$ **do**
  $\mathbf{B} \leftarrow$ SampleBatch($\mathbf{Z}$)
  $\widetilde{\mathbf{B}} \leftarrow$ Mask($\mathbf{B}$)
  $\mathcal{G} \leftarrow$ UpdateGradient(input $= \widetilde{\mathbf{B}}$, target $= \mathbf{B}$)
end for ▶ Prepare some non-overfitted trained weights
$[\mathbf{J}_1, \mathbf{J}_2], [\mathbf{Z}_1, \mathbf{Z}_2] \leftarrow$EvenlySplit($\mathbf{J}, \mathbf{Z}$)
$\mathcal{G}_1, \mathcal{G}_2 \leftarrow$MakeCopies($\mathcal{G}, 2$)
for $j = 1, 2$ **do**
  for $i = s_0 + 1, \ldots, s$ **do**
    $\mathbf{B} \leftarrow$ SampleBatch($\mathbf{Z}_j$)
    $\widetilde{\mathbf{B}} \leftarrow$ Mask($\mathbf{B}$)
    $\mathcal{G}_j \leftarrow$ UpdateGradient(input $= \widetilde{\mathbf{B}}$, target $= \mathbf{B}$)
    if ValidateAtStep(i) **then**
      $\ell \leftarrow$ ComputeLoss(input $=$ Mask($\mathbf{Z}_{1-j}$), target $= \mathbf{Z}_{1-j}$) ▶ Compute validation loss
      if MetEarlyStopCriterion(ℓ) **then**
        $i \leftarrow s$ ▶ Early stop
      end if
    end if
  end for
end for

Algorithm 2 Tabular data generation using TabTreeFormer

Input: Leaf index matrices $\mathbf{J}_1, \mathbf{J}_2$, data tokenizer $\mathcal{D}$, transformer-based language models $\mathcal{G}$, and number of rows generate n'
$n_1, n_2 \leftarrow \lfloor n'/2 \rfloor, \lceil n'/2 \rceil$ ▶ Assign number of rows for each network
Output: Synthetic tabular dataset $\mathbf{X}'$
for $j = 1, 2$ **do**
  $\mathbf{J}'_j \leftarrow$ Sample($\mathbf{J}_j, n_j$)
  $\mathbf{Z}'_j \leftarrow$ Mask(ToTokenIDs($\mathbf{J}'_j$)) ▶ Prepare tokens as prompts
  $\widetilde{\mathbf{X}}'_j \leftarrow \mathcal{G}$.Generate(partial_input $= \mathbf{Z}'_j$)
  $\mathbf{X}'_j \leftarrow \mathcal{D}$.Decode($\widetilde{\mathbf{X}}'_j$)
end for
$\mathbf{X}' \leftarrow$ Combine($\mathbf{X}'_1, \mathbf{X}'_2$)

A.2 Tree-based Model

A.2.1 Hyperparameter Tuning

For the tree-based classifier or regressor, we use Optuna (Akiba et al., 2019) to tune the hyperparameters of LightGBM (Ke et al., 2017) using the training data. If the dataset has a designated target column, which is the case for all our experimented datasets, the model is trained to predict that column. Otherwise, a random column can be the target. Then, fit the model with the selected hyperparameters for TabTreeFormer.

The hyperparameter space explored is

- **Learning Rate**: Logarithmic scale float in $[0.01, 0.3]$;
- **Number of Estimators**: Integers in $[50, 250]$ step 50;
- **Maximum Depth**: Integers in $[3, 10]$;
- **Number of Leaves**: Integers in $[20, 100]$ step 5;
- **Minimum Size per Leaf**: Integers in $[10, 50]$ step 5;
- **Feature Fraction**: Float in $[0.6, 1.0]$;
- **Bagging Fraction**: Float in $[0.6, 1.0]$;
- **L1 Regularization**: Float in $[0.0, 10.0]$;
- **L2 Regularization**: Float in $[0.0, 10.0]$.

Optimization was performed over 50 trials using 3-fold cross-validation, performance evaluated based on weighted F1 for classification and (negative) mean squared error for regression.

A.2.2 Leaf Index Matrix

In the subsequent steps of TabTreeFormer, leaf index matrix $\mathbf{J}$ of the real data $\mathbf{X}$ is computed. The indexing order of trees and their leaves used in the subsequent steps are directly taken from the fitted tree-based model. The leaf index matrix is essentially the result of `.apply` function for most tree-based models in `sklearn`.

Table 7: Hyperparameter configuration of TabTreeFormer of different sizes.

TabTreeFormer	S	L
Approx. #Params (in M)	5	40
hidden state dimensions	256	768
inner feed-forwared layer dimensions	1024	3072
number of attention heads per layer	8	12

A.3 Tokenization

For K-Means and quantile quantization, we use `sklearn.preprocessing.KBinsDiscretizer` with strategy "kmeans" and "quantile" respectively. For categorical values, we use `sklearn.preprocessing.OrdinalEncoder`.

In cases where the number of distinct values expected to be kept is way larger than typical natural language models' vocabulary size, two or more consecutive quantile quantizers may be used, where subsequent quantizers are used to quantize the values in the same quantile as suggested by previous quantizers. Thus, with q quantizers, a total of Q^q values can be represented, which is sufficient to cover most practical cases. In this paper, we present $Q = 1000$ as sufficient precision for simplicity.

A.4 Transformer

A.4.1 Model Configuration

For the language model, we use Distill-GPT2 (Sanh et al., 2019), based on the GPT2 architecture (Radford et al., 2019), with several modifications. The vocabulary size and maximum sequence length are adjusted based on the dataset, and the values are likely much lower than the requirement for natural language models. For example, if $n_l = 200, n_c = 20, n_b = K = 10, n_q = Q = 1000$, then $V = 1233$, and if $T = 200, m_d = 10, m_c = 10$, then $L = 222$, while natural language models typically have $V > 30000, L > 1000$.

In this paper, we present 3 versions of TabTreeFormer of different sizes. Table 7 shows their details. The model sizes are obtained for diabetes (Smith et al., 1988) dataset. Actual values may vary based on the fitted tree-based model and dataset.

A.4.2 Token Sequence Construction

The tokens in Table 1 are conceptual tokens. When converting the leaf, category, K-Means bin, and quantile IDs into token IDs in actual implementation, we do not explicitly make the textual tokens, but instead, directly add corresponding token type's offset.

A.4.3 Masking

Instead of using a fixed mask ratio, we apply varying mask ratios to enhance the diversity of training data. Tree-related tokens (yellow section in Fig. 2) are masked with a ratio sampled uniformly from $[0.5, 0.75]$, while actual value tokens (blue section in Fig. 2) are masked with a ratio sampled uniformly from $[0.25, 0.5]$. In the no-mask (NM) version, we set both mask ratios to be constant 0.

To ensure that the quantile token is always masked if its corresponding K-Means bin token is masked within the same numeric column, we first generate a random mask. If a violation is detected (i.e., the K-Means bin token is unmasked while the quantile token is masked), we swap the mask values of these two positions. This approach ensures that masks are generated both efficiently and accurately.

A.4.4 Training Configuration

The reduced model size allows for a significantly larger batch size compared to typical natural language models. We use a batch size of 128 per device, reduce by half in case of CUDA OOM[‡], and train for up to 5,000 steps ($s = 5,000$ in Algorithm 1) with a learning rate of 5×10^{-4} in FP16 precision, utilizing the Hugging Face Trainer. Early stopping may be triggered by validation loss with a patience of 3 every 100 steps. Patience is set to 100 in the NM version. The initial training on both splits (Line 9-13 in Algorithm 1) is trained with the number of steps (s_0) set to the smaller of the number of steps required for 20 epochs and 1/10 of s.

A.4.5 Model Inference

Tree-leaf tokens from real data after masking are used as prompts for generation. Values of categorical columns are more likely to be memorized than numeric values by our ordinal learning method. Therefore, we apply a higher temperature to the tokens for categorical values and a lower one to the tokens for numeric values. In particular, we set the temperature for categorical tokens to be 2.0 and numeric tokens, including bin and qunatile tokens, to be 1.0. In the NM version, we set the temperatures to be 0.2 and 0.1 respectively.

Recall that a set of valid tokens can be pre-determined at each position. Formally, let p_i be the corresponding feature index for the i-th token from data tokenizer, and R_i indicates the type of this token. For example, in Fig. 2, $p_1 = 1, p_2 = 1, p_3 = 2, R_1 = \text{bin}, R_2 = \text{quant}, R_3 = \text{cat}$. Then, let $\mathcal{V}_i$ be the set of tokens allowed at position $i \in$

[‡]Among all $10 \times 3 = 30$ experiments, TabTreeFormer-L hits CUDA OOM in one experiment, and no other TabTreeFormer experiments had memory issues with this batch size.

$\{1, 2, \ldots, L\}$, we have Equation 7.

$$\mathcal{V}_i = \begin{cases} \{[\text{BOS}]\}, & i = 1, \\ \{[\text{leaf1}], \ldots, [\text{leaf}l_{i-1}]\}, & 2 \le i \le T+1, \\ \{[\text{bin1}], \ldots, [\text{bin}b_{p_{i-T-1}}]\}, & T+2 \le i \le L-1,\ R_{i-T-1} = \text{bin}, \\ \{[\text{quant1}], \ldots, [\text{quant}q_{p_{i-T-1}}]\}, & T+2 \le i \le L-1,\ R_{i-T-1} = \text{quant}, \\ \{[\text{cat1}], \ldots, [\text{cat}b_{p_{i-T-1}}]\}, & T+2 \le i \le L-1,\ R_{i-T-1} = \text{cat}, \\ \{[\text{EOS}]\}, & i = L \end{cases} \tag{7}$$

Valid tokens for a specific position is enforced by setting the logits of invalid tokens at the position to negative infinity before sampling.

B Supplementary Quantile Embeddings Explanation and Analysis

B.1 Preliminaries: Function-Generated Positional Embeddings

While positional embeddings can also be a learned embedding like word tokens, Vaswani et al. (2017) found that its performance is not necessarily significantly better than function-generated positional embeddings. Moreover, function-generated embeddings allow the model to extrapolate positional information in sentences longer than those in the training corpora.

The typical functions to generate positional embeddings written in a similar format as Equation 3 (expressed in i and d) are shown in Equation 8 (Vaswani et al., 2017).

$$PE_{id} = \begin{cases} \sin\left(\frac{i}{10000^{\frac{d}{D}}}\right) & \text{if } 2 \mid i \\ \cos\left(\frac{i}{10000^{\frac{d-1}{D}}}\right) & \text{if } 2 \nmid i \end{cases} \tag{8}$$

The functions are essentially sine and cosine functions, where the frequency is controled by the dimension indices. Using these periodic functions is particularly suitable for positional embeddings, which focus more on relative differences than absolute differences.

B.2 Choice of the Core Generator Function for Quantile Embeddings

Periodic functions are proper generator functions for positional embeddings because of the relative nature of positions. Similarly, due to the absolute and monotonic nature of quantile values, monotonic functions are more suitable for the generator functions for quantile embeddings.

The simplest of monotonic functions are linear functions, but they suffer from the following problems:

- To obtain different linear functions at different embedding dimensions, the slopes and intercepts need to be changed. However, in order for the embedded values to fall in a reasonable range (no extremely large or small values), the choice of proper slopes and intercepts becomes very limited, which hurts the diversity between different embedding dimensions. This significantly downgrades the effect of having different embedding dimensions.

- Having linear functions in all dimensions means that different dimensions are in strict linear relations to one another. Note that transformers are heavily dependent on linear projections. In particular, the input embeddings are linearly projected to Q, K, V vectors of the first attention layer. Therefore, universal linear relations among all embedded dimensions would degenerate the power of the first few layers of the network, which is an undesired outcome.

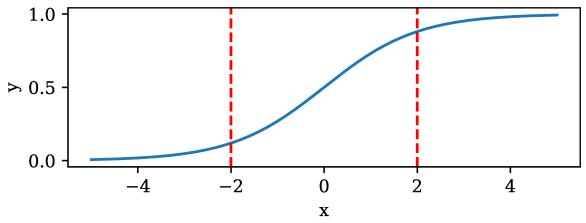

Figure 12: Sigmoid function $y = \text{sigmoid}(x)$. The red vertical dashed lines highlight the range $x \in [-2, 2]$ where the changes of values are considered non-trivial.

Therefore, besides being monotonic, we also require the generator function to be non-linear and have a controled range (so ideally bounded). In this paper, we choose sigmoid, a simple function that satisfies the non-linear monotonic and bounded requirements. Nevertheless, we also note that other functions satisfying the requirements may also be used, such as tanh, softsign, and piece-wise linear functions.

B.3 Construction of Quantile Embeddings

For the standard sigmoid function $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$, we consider the range $[-2, 2]$ as the core range of x that has non-trivial changes of values (see Fig. 12). To project the total Q quantile tokens on this core range, we need to scale and shift the quantile ID i to $4\left(\frac{i}{Q} - \frac{1}{2}\right)$ (recall Equation 3).

The scale factor S_d (recall Equation 1) controls how the core range is distributed over the total Q quantile tokens. For a given scale factor value $s \in \mathbb{N}^+$, we assign $2s$ embedded dimensions with different offsets. More dimensions are assigned to larger scale factors because of the smaller number of tokens having transformed input values falling in the core range of the sigmoid function. Thus, the vector of S_d for $d = 0, 1, \ldots, D-1$ is constructed by

`repeat(arange(n)+1, 2*(arange(n)+1))[:D]` of a large enough `n`.

Proposition 1. *Equation 1 is a closed form of the scale factor construction above (*`repeat(arange(n)+1, 2*(arange(n)+1))[:D]`*).*

Proof. The construction means that for scale factor value $s \in \mathbb{N}^+$ starting from 1, the same value repeats $2s$ times. Let S'_d be the value by the construction, and $\mathcal{J}(s) = \{d \mid S'_d = s\}$, then $\mathcal{J}(s)$ must be a set of consecutive integers, and

$$\min \mathcal{J}(s) = \sum_{i=1}^{s-1} 2s = (1+s-1)(s-1) = s^2 - s \quad (9)$$

$$\max \mathcal{J}(s) = \min \mathcal{J}(s) + 2s - 1 = s^2 + s - 1 \quad (10)$$

Let $\hat{S}_d = \frac{1+\sqrt{1+4d}}{2}$, then $S_d = \lfloor \hat{S}_d \rfloor$. Then, by $s \geq 1$,

$$\begin{aligned}
\hat{S}_{\min \mathcal{J}(s)} &= \hat{S}_{s^2-s} \\
&= \frac{1+\sqrt{1+4\left(s^2-s\right)}}{2} \\
&= \frac{1+\sqrt{(2s-1)^2}}{2} \\
&= \frac{1+|2s-1|}{2} \\
&= s, \qquad (3a) \\
\hat{S}_{\max \mathcal{J}(s)} &= \hat{S}_{s^2+s-1} \\
&= \frac{1+\sqrt{1+4\left(s^2+s-1\right)}}{2} \\
&< \frac{1+\sqrt{(2s+1)^2}}{2} \\
&= \frac{1+|2s+1|}{2} \\
&= s+1. \qquad (3b)
\end{aligned}$$

Note that $\hat{S}_d$ is monotonically increasing with respect to d, then we must have

$$\hat{S}_d \in [s, s+1),\ \forall d \in \mathcal{J}(s) \quad (11)$$

This concludes to

$$S_d = \lfloor \hat{S}_d \rfloor = s \quad (12)$$

Given $s \in \mathbb{N}^+$, then $S_d \in \mathbb{N}^+$. □

For embedding dimensions corresponding to the same scale factor S_d, we apply evenly distributed offsets in $[-2S_d, 2S_d]$, where the constant factors $-2, 2$ also come from the core range $x \in [-2, 2]$. Thus, the vector of offsets O_d for $d = 0, 1, \ldots, D-1$ is constructed by $concat(linspace(-2{*}x, 2{*}x, 2{*}x)$ for x in $(arange(n)+1))[:D]|$.

Proposition 2. *Equation 2 is a closed form of the offset construction above (*$concat(linspace(-2*x, 2*x, 2*x)$ *for* x *in* $(arange(n)+1))[:D]|$*).*

Proof. The `x` in the construction is s in the previous proof. Let the i-th offset value with $S_d = s$ be o_{si} by the construction. Then we have $i \in [0, 2s)$ and

$$\begin{aligned}
o_{si} &= -2s + i \cdot \frac{2s-(-2s)}{2s-1} \\
&= \frac{-4s^2+2s+4si}{2s-1} \\
&= \frac{2s\left(-2s+(2i+1)\right)}{2s-1} \qquad (13)
\end{aligned}$$

Let O'_d be the d-th value in the constructed offset value, regardless of scale factor, then substitute with Equation 13, 9 and 2,

$$O'_d = o_{S_d(d-\min \mathcal{J}(S_d))} = \frac{2S_d(-2S_d + (2(d-S_d^2+S_d)+1))}{2S_d-1} \quad (14)$$

$$= \frac{-4S_d^3 + (4d+2)S_d}{2S_d-1} = O_d \quad (15)$$

Furthermore, $o_{si} \in [-2s, 2s]$, so $O_d \in [-2S_d, 2S_d]$. □

Applying the sigmoid function on the quantile IDs with the scale factors and offsets, we obtain the quantile embedding value generators in Equation 3.

B.4 Comparison between Quantile Embeddings and Positional Embeddings

Table 8 shows a summary of the comparison between the proposed quantile embeddings and the typical function-generated positional embeddings. The two function-generated embeddings are also visualized in Fig. 13.

B.5 Proof of Theorem 2

Proof. For simplicity of symbols, let $q_{id} = QE_{id}$. By Equation 3, $q_{id} = \text{sigmoid}(k_d i + b_d)$ for some $k_d \in \mathbb{R}^+, b_d \in \mathbb{R}$. Note that sigmoid is strictly monotonically increasing, and the linear function $k_d i + b_d$ with $k_d > 0$ is also strictly monotonically increasing concerning i. Thus, q_{id} is strictly monotonically increasing with respect to i.

If $|i-j| < |i-k|$, then $|q_{id} - q_{jd}| < |q_{id} - q_{kd}|, \forall d \in \{0, 1, \ldots, D-1\}$, by the monotonicity of q_{id} with respect

Table 8: Comparison between the proposed function-generated quantile and typical function-generated positional embeddings for transformers (Vaswani et al., 2017).

	Quantile Embeddings	Positional Embeddings
Core function	Sigmoid	Trigonometric
Core relation	Absolute ordinal value	Relative distance
Function property	Non-linear and monotonic, range-controled	Periodic, range-controled
Dimensions differ in	Scale factor and offset	Frequency and function (sin vs. cos)

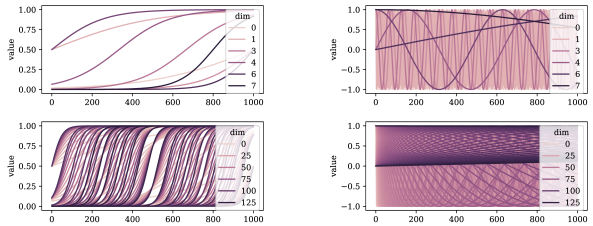

Figure 13: Visualized comparison of function-generated quantile and positional embeddings. The x-axis represents $i \in \{0, 1, \ldots, Q-1\}$ (where $Q = 1000$) and y-axis represents the corresponding embedded values. Different colors of lines represent different embedded dimensions. **Left:** quantile embeddings (Equation 3), **Right:** positional embeddings (Equation 8), **Top:** $D = 8$, **Bottom:** $D = 128$.

to i and triangle inequality. Then, by the strictly increasing monotonicity of $f(x) = x^p$ and $f(x) = x^{\frac{1}{p}}$ for $p \geq 1$ on $x \in \mathbb{R}^+ \cup \{0\}$ and by the monotonicity of the sum of monotonic functions, we must have

$$\begin{aligned}\|\mathbf{q}_i - \mathbf{q}_j\|_p &= \Big(\sum_{d=0}^{D-1} |q_{id} - q_{jd}|^p\Big)^{\frac{1}{p}} \\ &< \Big(\sum_{d=0}^{D-1} |q_{id} - q_{kd}|^p\Big)^{\frac{1}{p}} \\ &= \|\mathbf{q}_i - \mathbf{q}_k\|_p \end{aligned} \tag{16}$$

and by the monotonicity of the maximum of monotonic functions, we must have

$$\begin{aligned}\|\mathbf{q}_i - \mathbf{q}_j\|_\infty &= \max_{0\leq d<D} |q_{id} - q_{jd}| \\ &< \max_{0\leq d<D} |q_{id} - q_{kd}| \\ &= \|\mathbf{q}_i - \mathbf{q}_k\|_\infty \end{aligned} \tag{17}$$

Therefore, if $|i-j| < |i-k|$, then $\|\mathbf{q}_i - \mathbf{q}_j\|_p < \|\mathbf{q}_i - \mathbf{q}_k\|_p$.

In the inverse direction, we prove the contrapositive. If $|i-j| \geq |i-k|$, then $|q_{id} - q_{jd}| \geq |q_{id} - q_{kd}|, \forall d \in \{0, 1, \ldots, D-1\}$. Next, by similar monotonic arguments above, we must have

$$\begin{aligned}\|\mathbf{q}_i - \mathbf{q}_j\|_p &= \Big(\sum_{d=0}^{D-1} |q_{id} - q_{jd}|^p\Big)^{\frac{1}{p}} \\ &\geq \Big(\sum_{d=0}^{D-1} |q_{id} - q_{kd}|^p\Big)^{\frac{1}{p}} \\ &= \|\mathbf{q}_i - \mathbf{q}_k\|_p \end{aligned} \tag{18}$$

and

$$\begin{aligned}\|\mathbf{q}_i - \mathbf{q}_j\|_\infty &= \max_{0\leq d<D} |q_{id} - q_{jd}| \\ &\geq \max_{0\leq d<D} |q_{id} - q_{kd}| \\ &= \|\mathbf{q}_i - \mathbf{q}_k\|_\infty \end{aligned} \tag{19}$$

The above shows that if $|i-j| \geq |i-k|$, then $\|\mathbf{q}_i - \mathbf{q}_j\|_p \geq \|\mathbf{q}_i - \mathbf{q}_k\|_p$. Therefore, if $\|\mathbf{q}_i - \mathbf{q}_j\|_p < \|\mathbf{q}_i - \mathbf{q}_k\|_p$, then $|i-j| < |i-k|$.

Summarizing the above, we have $|i-j| < |i-k|$ if and only if $\|\mathbf{q}_i - \mathbf{q}_j\|_p < \|\mathbf{q}_i - \mathbf{q}_k\|_p$, and the conclusion holds for any p. □

Remark 1. *According to the proof, any strictly monotonically increasing core function, not restricted to sigmoid, makes Theorem 2 hold.*

Remark 2. *Any monotonically increasing core function, including non-strictly monotonic ones, makes the "if" direction of Theorem 2 hold, with no guarantee of the "only if" direction (the two p-norm distances can be equal).*

C Supplementary Loss Function Explanation and Analysis

C.1 Proof of Lemma 1

Proof. Firstly, note that $\frac{\partial w_{ti}}{\partial z_j} = 0, \forall i, j \in \{1, \ldots, V\}$, by independence between the two. Also mind that $w_{ti} > 0, \forall i \in \{1, \ldots, V\}$.

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{oce}}(\mathbf{z}, t)}{\partial z_t} &= -\frac{\sum_{i=1}^V w_{ti} e^{z_i}}{w_{tt} e^{z_t}} \\ &\quad \cdot \frac{w_{tt} e^{z_t} \sum_{i=1}^V w_{ti} e^{z_i} - w_{tt} e^{z_t} \cdot w_{tt} e^{z_t}}{\left(\sum_{i=1}^V w_{ti} e^{z_i}\right)^2} \\ &= -\frac{\sum_{i=1}^V w_{ti} e^{z_i} - w_{tt} e^{z_t}}{\sum_{i=1}^V w_{ti} e^{z_i}} \qquad (20) \\ &< 0, \qquad (21) \end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{oce}}(\mathbf{z}, t)}{\partial z_i} &= -\frac{\sum_{j=1}^V w_{tj} e^{z_j}}{w_{tt} e^{z_t}} \cdot \frac{-w_{ti} e^{z_i} \cdot w_{tt} e^{z_t}}{\left(\sum_{j=1}^V w_{tj} e^{z_j}\right)^2} \\ &= \frac{w_{ti} e^{z_i}}{\sum_{j=1}^V w_{tj} e^{z_j}} > 0. \qquad (22) \end{aligned}$$

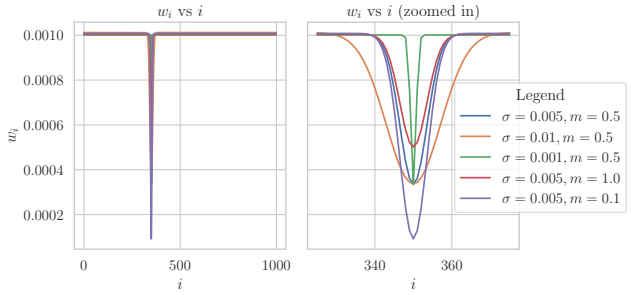

Figure 14: The relation of weights w_{ti} (normalized so that $\|\mathbf{w}\| = 1$ for visualization purpose to token index i, with vocabulary size $V = 1000$, and target token $t = 350$.

Thus, $\mathcal{L}_{\text{oce}}(\mathbf{z}, t)$ is optimized (minimized) when $z_t \to \infty, z_i \to -\infty (t \neq i)$. □

C.2 Proof of Theorem 3

Proof. Express $\mathcal{L}_{\text{oce}}$ and $\mathcal{L}_{\text{ce}}$ using $\mathbf{p}$, we have

$$\mathcal{L}_{\text{ce}}(\mathbf{z}, t) = -\log p_t, \tag{23}$$

$$\begin{aligned}
\mathcal{L}_{\text{oce}}(\mathbf{z}, t) &= -\log \frac{w_{tt} e^{z_t}}{\sum_{i=1}^{V} w_{ti} e^{z_i}} \\
&= -\log \frac{w_{tt} \dfrac{e^{z_t}}{\sum_{i=1}^{V} e^{z_i}}}{\sum_{i=1}^{V} w_{ti} \dfrac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}}} \\
&= -\log \frac{w_{tt} p_t}{\sum_{i=1}^{V} w_{ti} p_i} \\
&= -\log \frac{w_{tt} p_t}{D(\mathbf{z})}.
\end{aligned} \tag{24}$$

By $\mathcal{L}_{\text{ce}}(\mathbf{z}, t) = \mathcal{L}_{\text{ce}}(\mathbf{z}^*, t)$, we have $p_t = p_t^*$. Given $D(\mathbf{z}) < D(\mathbf{z}^*)$, we must have $\mathcal{L}_{\text{oce}}(\mathbf{z}, t) < \mathcal{L}_{\text{oce}}(\mathbf{z}^*, t)$.

□

C.3 Detailed Explanations and Analysis of the Weight Function (Equation 6)

Recall the function of weight given the target token and current token in Equation 6. This section will provide detailed explanations and analysis of it.

The exponential term comes from the intuition that the difference between closer quantiles is much more significant than the difference between farther quantiles. Some scaling and shifting is applied on the exponential term.

Fig. 14 illustrates the relationship between w_{ti} and i under varying values of σ and m. The parameter σ primarily governs the width around the target token where the weights are significantly altered; larger σ values result in a broader width. In contrast, m dictates the intensity of the weight modification, with smaller m values leading to more pronounced changes in weight.

C.4 Overall Loss Computation

Concerning the combination of valid quantile tokens and other non-quantile tokens, let valid quantile tokens have IDs in the range $[Q_s, Q_e)$. The valid token group loss (for quantile tokens at this position) is defined by Equation 25 when $t \in [Q_s, Q_e)$.

$$\mathcal{L}_{\text{vg}}(\mathbf{z}, t) = -\log \frac{\sum_{i \in [Q_s, Q_e)} e^{z_i}}{\sum_{i=1}^{V} e^{z_i}} \tag{25}$$

In sum, Equation 26 expresses the loss function on one token.

$$\mathcal{L}(\mathbf{z}, t) = \begin{cases} \mathcal{L}_{\text{ce}}(\mathbf{z}, t) & \text{if } t \notin [Q_s, Q_e) \\ \mathcal{L}_{\text{oce}}(\mathbf{z}_{Q_s:Q_e}, t - Q_s) + \mathcal{L}_{\text{vg}}(\mathbf{z}, t) & \text{if } t \in [Q_s, Q_e) \end{cases} \tag{26}$$

Although the ordinal property also holds for K-Means bin IDs, we do not calculate the loss on them this way because the number of different bins is small, and we expect the model to predict the bins accurately.

D Experiment Setup Details

D.1 Experiment Environment

Experiments were conducted on a system running Ubuntu 22.04.1 LTS with kernel version 6.8.0. The machine featured a 13th Gen Intel Core i9-13900K CPU with 24 cores and 32 threads, capable of a maximum clock speed of 5.8 GHz. GPU computations were performed using a single NVIDIA GeForce RTX 4090 with 24 GB VRAM. The environment was set up with NVIDIA driver version 565.57.01 and CUDA 12.7.

D.2 Details on Datasets

Table 9 shows the details of the datasets we used. Datasets are downloaded by `sklearn.datasets.fetch_openml`, with the dataset name as input. Random splits of $8:2$ of training and test sets are applied.

D.3 Baseline Choices and Implementation

For i) non-neural-network method (tree-based sampling), ii) GAN (Goodfellow et al., 2014), iii) VAE (Kingma and Welling, 2014), and iv) Diffusion (Ho et al., 2020), we choose one recent, representative, and well-performing (especially on MLE) model each: Forest Diffusion (Jolicoeur-Martineau et al., 2024), CTAB-GAN+ (Zhao et al., 2024),

Table 9: Datasets used in experiments. #R, #F, #N, and #C represents the number of rows, features (including target column), numeric features, and categorical features respectively. The task can be classification, denoted "clf" followed by the number of classes in the bracket, and regression, denoted "reg". Aliases in bracket in "Name" column show the names used in this paper.

Name	#R	#F	#N	#C	Task
adult Kohavi (1996)	48842	15	2	13	clf(2)
bank-marketing (bank) Moro et al. (2011)	45211	17	7	10	clf(2)
boston Harrison and Rubinfeld (1978)	506	14	12	2	reg
breast-w (breast) Wolberg and Mangasarian (1990)	699	10	9	1	clf(2)
credit-g (credit) Hofmann (1994)	1000	21	7	14	clf(2)
diabetes Smith et al. (1988)	768	9	8	1	clf(2)
iris Fisher (1936)	150	5	4	1	clf(3)
qsar-biodeg (qsar) Mansouri et al. (2013)	1055	42	41	1	clf(2)
wdbc Street et al. (1993)	569	31	30	1	clf(2)

TTVAE (Wang and Nguyen, 2025), and TabSyn (Zhang et al., 2024). For auto-regressive and masked transformers, which are more aligned with our work, we include more works: GReaT (Borisov et al., 2023), REaLTabFormer (Solatorio and Dupriez, 2023), TabMT (Gulati and Roysdon, 2023), TabuLa (Zhao et al., 2023). All of them will use a DistilGPT2 (Sanh et al., 2019) as the transformer backbone except for TabMT (Gulati and Roysdon, 2023), which has specific settings described in its paper.

Other large language model (LLM)-based methods, such as HARMONIC (Wang et al., 2024), are excluded from our comparison for fairer comparison because

- They usually require a good-quality dataset description, but we want to focus on the case without additional information besides the data per se.
- Their performance is dependent on a very large and good LLM. The improvement of the LLM model may be more important than the improvement of the tabular-specific design, but we want to showcase the effectiveness of tabular-specific designs instead of an LLM.

We use the Synthcity (Qian et al., 2023) implementation for GReaT (Borisov et al., 2023). Our benchmark code is adapted from Synthcity (Qian et al., 2023) too. We also include CTAB-GAN+ (Zhao et al., 2024), REaLTabFormer (Solatorio and Dupriez, 2023), TabSyn (Zhang et al., 2024), Forest Diffusion (Jolicoeur-Martineau et al., 2024), TabuLa (Zhao et al., 2023), and TTVAE (Wang and Nguyen, 2025) using their official code on GitHub or public SDK in the benchmark. TabMT (Gulati and Roysdon, 2023) has no publicly available code, so we do experiments based on a reproduction of the model based on the paper's description, so the model could be slightly different from that paper.

D.4 Machine Learning Efficacy (MLE) Metrics Implementation

Instead of running all models with fixed, typically default, parameters across all datasets, we perform hyperparameter tuning before reporting their performance. The tuning process follows the same methodology used to optimize the core tree-based model in TabTreeFormer, but with 30 trials per round of optimization.

For linear models, no hyperparameter tuning is performed, as they follow the classical definition without additional tunable parameters.

The hyperparameter space explored for random forest is

- **Number of Estimators**: Integers in $[100, 300]$ step 50;
- **Maximum Depth**: Integers in $[5, 20]$ step 5;
- **Minimum Samples Split**: Integers in $[2, 10]$ step 2;
- **Minimum Size per Leaf**: Integers $[1, 5]$;
- **Maximum Features**: Value in "sqrt", "log2", and NULL;
- **Bootstramp**: Enabled or disabled.

The hyperparameter space explored for XGBoost is:

- **Learning Rate**: Logarithmic scale float in $[0.01, 0.3]$;
- **Number of Estimators**: Integers in $[100, 300]$ step 50;
- **Maximum Depth**: Integers in $[3, 10]$;
- **Minimum Child Weight**: Logarithmic scale float in $[1.0, 10.0]$;
- **Minimum Split Loss Gamma**: Float in $[0.0, 0.5]$;

Table 10: Performance comparison between different models in terms of MLE. Classification tasks are evaluated by weighted AUC ROC scores, and regression tasks are evaluated by R^2 scores. The last row shows the average relative error versus real. The best scores and the second best scores are highlighted in bold with and without underscore, respectively. Equal values in the first 3 digits may be compared by the 4-th or 5-th digits.

Dataset	ML	Real	CTAB+	TTVAE	TabSyn	FD	GReaT	RTF	TabMT	TabuLa	TTF-S	TTF-L	TTF-NP
adult	LN	0.914	$0.904_{\pm 0.001}$	$0.875_{\pm 0.010}$	$0.910_{\pm 0.002}$	$\mathbf{0.911}_{\pm 0.000}$	$0.911_{\pm 0.000}$	$0.911_{\pm 0.000}$	$0.911_{\pm 0.000}$	$\underline{\mathbf{0.913}}_{\pm 0.000}$	$0.906_{\pm 0.001}$	$0.911_{\pm 0.001}$	$0.910_{\pm 0.000}$
	RF	0.910	$0.902_{\pm 0.002}$	$0.869_{\pm 0.011}$	$0.905_{\pm 0.003}$	$\mathbf{0.908}_{\pm 0.000}$	$0.905_{\pm 0.000}$	$\underline{\mathbf{0.908}}_{\pm 0.000}$	$0.907_{\pm 0.002}$	$0.888_{\pm 0.002}$	$0.901_{\pm 0.001}$	$0.905_{\pm 0.000}$	$0.903_{\pm 0.002}$
	XGB	0.914	$0.905_{\pm 0.001}$	$0.870_{\pm 0.012}$	$0.910_{\pm 0.003}$	$\mathbf{0.912}_{\pm 0.001}$	$0.911_{\pm 0.000}$	$\underline{\mathbf{0.912}}_{\pm 0.001}$	$0.911_{\pm 0.001}$	$0.899_{\pm 0.005}$	$0.906_{\pm 0.001}$	$0.911_{\pm 0.001}$	$0.909_{\pm 0.001}$
bank	LN	0.907	$0.901_{\pm 0.003}$	$0.885_{\pm 0.004}$	$0.900_{\pm 0.005}$	$0.903_{\pm 0.001}$	$\underline{\mathbf{0.907}}_{\pm 0.000}$	$0.904_{\pm 0.000}$	$0.862_{\pm 0.005}$	$0.906_{\pm 0.001}$	$0.906_{\pm 0.000}$	$\mathbf{0.906}_{\pm 0.000}$	$0.901_{\pm 0.003}$
	RF	0.930	$0.902_{\pm 0.002}$	$0.884_{\pm 0.009}$	$0.907_{\pm 0.006}$	$0.917_{\pm 0.001}$	$0.907_{\pm 0.001}$	$0.913_{\pm 0.001}$	$0.886_{\pm 0.011}$	$\underline{\mathbf{0.923}}_{\pm 0.004}$	$0.913_{\pm 0.002}$	$0.918_{\pm 0.001}$	$\mathbf{0.921}_{\pm 0.003}$
	XGB	0.936	$0.906_{\pm 0.000}$	$0.877_{\pm 0.011}$	$0.913_{\pm 0.007}$	$0.920_{\pm 0.001}$	$0.910_{\pm 0.001}$	$0.918_{\pm 0.002}$	$0.892_{\pm 0.007}$	$0.925_{\pm 0.003}$	$0.919_{\pm 0.002}$	$\mathbf{0.925}_{\pm 0.003}$	$\underline{\mathbf{0.926}}_{\pm 0.002}$
boston	LN	0.590	$0.264_{\pm 0.034}$	$0.000_{\pm 0.079}$	$0.533_{\pm 0.036}$	$0.542_{\pm 0.016}$	$0.397_{\pm 0.000}$	$0.529_{\pm 0.028}$	$0.409_{\pm 0.015}$	$0.484_{\pm 0.069}$	$0.525_{\pm 0.035}$	$\mathbf{0.575}_{\pm 0.011}$	$\underline{\mathbf{0.591}}_{\pm 0.000}$
	RF	0.662	$0.129_{\pm 0.015}$	$0.124_{\pm 0.068}$	$0.658_{\pm 0.025}$	$0.608_{\pm 0.008}$	$0.281_{\pm 0.001}$	$0.584_{\pm 0.006}$	$0.439_{\pm 0.096}$	$0.589_{\pm 0.050}$	$0.597_{\pm 0.038}$	$\underline{\mathbf{0.694}}_{\pm 0.041}$	$\mathbf{0.683}_{\pm 0.016}$
	XGB	0.701	$0.097_{\pm 0.078}$	$0.211_{\pm 0.121}$	$0.647_{\pm 0.017}$	$0.604_{\pm 0.024}$	$0.277_{\pm 0.014}$	$0.578_{\pm 0.056}$	$0.446_{\pm 0.115}$	$0.615_{\pm 0.069}$	$0.594_{\pm 0.061}$	$\mathbf{0.665}_{\pm 0.026}$	$\underline{\mathbf{0.691}}_{\pm 0.023}$
breast	LN	0.985	$0.913_{\pm 0.091}$	$0.963_{\pm 0.006}$	$\mathbf{0.987}_{\pm 0.001}$	$0.986_{\pm 0.002}$	$0.985_{\pm 0.000}$	$\underline{\mathbf{0.987}}_{\pm 0.002}$	$0.983_{\pm 0.004}$	$0.985_{\pm 0.001}$	$0.981_{\pm 0.005}$	$0.987_{\pm 0.002}$	$0.984_{\pm 0.004}$
	RF	0.985	$0.968_{\pm 0.012}$	$0.979_{\pm 0.004}$	$0.981_{\pm 0.002}$	$0.975_{\pm 0.005}$	$0.979_{\pm 0.004}$	$0.980_{\pm 0.000}$	$0.978_{\pm 0.002}$	$0.976_{\pm 0.005}$	$0.978_{\pm 0.001}$	$\mathbf{0.983}_{\pm 0.002}$	$\underline{\mathbf{0.984}}_{\pm 0.002}$
	XGB	0.984	$0.958_{\pm 0.025}$	$0.970_{\pm 0.009}$	$\underline{\mathbf{0.987}}_{\pm 0.001}$	$\mathbf{0.984}_{\pm 0.002}$	$0.982_{\pm 0.002}$	$0.982_{\pm 0.002}$	$0.982_{\pm 0.002}$	$0.979_{\pm 0.002}$	$0.982_{\pm 0.002}$	$0.983_{\pm 0.003}$	$0.982_{\pm 0.003}$
credit	LN	0.836	$0.538_{\pm 0.093}$	$0.500_{\pm 0.000}$	$0.780_{\pm 0.017}$	$\mathbf{0.827}_{\pm 0.010}$	$0.665_{\pm 0.000}$	$0.801_{\pm 0.006}$	$0.773_{\pm 0.038}$	$0.814_{\pm 0.016}$	$0.816_{\pm 0.020}$	$0.755_{\pm 0.011}$	$\underline{\mathbf{0.834}}_{\pm 0.001}$
	RF	0.837	$0.505_{\pm 0.097}$	$0.500_{\pm 0.000}$	$0.791_{\pm 0.007}$	$\mathbf{0.819}_{\pm 0.009}$	$0.722_{\pm 0.014}$	$\underline{\mathbf{0.820}}_{\pm 0.010}$	$0.763_{\pm 0.019}$	$0.805_{\pm 0.019}$	$0.781_{\pm 0.026}$	$0.741_{\pm 0.027}$	$0.813_{\pm 0.026}$
	XGB	0.844	$0.509_{\pm 0.088}$	$0.500_{\pm 0.000}$	$0.795_{\pm 0.014}$	$\underline{\mathbf{0.836}}_{\pm 0.010}$	$0.739_{\pm 0.006}$	$0.797_{\pm 0.039}$	$0.761_{\pm 0.030}$	$0.817_{\pm 0.020}$	$0.796_{\pm 0.011}$	$0.764_{\pm 0.014}$	$\mathbf{0.826}_{\pm 0.026}$
diabetes	LN	0.884	$0.817_{\pm 0.028}$	$0.880_{\pm 0.003}$	$0.883_{\pm 0.006}$	$0.885_{\pm 0.001}$	$0.858_{\pm 0.000}$	$0.878_{\pm 0.003}$	$0.855_{\pm 0.014}$	$0.876_{\pm 0.007}$	$\underline{\mathbf{0.890}}_{\pm 0.010}$	$\mathbf{0.886}_{\pm 0.001}$	$0.843_{\pm 0.058}$
	RF	0.869	$0.802_{\pm 0.014}$	$0.849_{\pm 0.004}$	$0.844_{\pm 0.012}$	$0.856_{\pm 0.012}$	$0.824_{\pm 0.012}$	$0.841_{\pm 0.010}$	$0.845_{\pm 0.009}$	$\underline{\mathbf{0.863}}_{\pm 0.003}$	$0.855_{\pm 0.014}$	$\mathbf{0.859}_{\pm 0.010}$	$0.814_{\pm 0.073}$
	XGB	0.867	$0.811_{\pm 0.005}$	$0.851_{\pm 0.001}$	$0.847_{\pm 0.009}$	$0.846_{\pm 0.012}$	$0.831_{\pm 0.003}$	$0.849_{\pm 0.010}$	$0.840_{\pm 0.009}$	$\underline{\mathbf{0.857}}_{\pm 0.009}$	$\mathbf{0.852}_{\pm 0.015}$	$0.848_{\pm 0.017}$	$0.811_{\pm 0.085}$
iris	LN	1.000	$0.272_{\pm 0.068}$	$0.996_{\pm 0.006}$	$0.999_{\pm 0.002}$	$0.997_{\pm 0.005}$	$0.985_{\pm 0.000}$	$0.983_{\pm 0.015}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$		$0.971_{\pm 0.018}$	$0.997_{\pm 0.005}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$
	RF	1.000	$0.396_{\pm 0.099}$	$0.992_{\pm 0.012}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$0.993_{\pm 0.005}$	$0.993_{\pm 0.005}$	-	$0.999_{\pm 0.002}$	$0.988_{\pm 0.017}$	$0.967_{\pm 0.047}$
	XGB	1.000	$0.211_{\pm 0.042}$	$0.999_{\pm 0.002}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$0.998_{\pm 0.003}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$		$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$0.999_{\pm 0.001}$
qsar	LN	0.906	$0.716_{\pm 0.085}$	$0.669_{\pm 0.120}$	$0.867_{\pm 0.012}$	$0.881_{\pm 0.005}$	$0.673_{\pm 0.000}$	$0.880_{\pm 0.003}$	$\mathbf{0.887}_{\pm 0.005}$	$0.868_{\pm 0.034}$	$0.874_{\pm 0.013}$	$0.876_{\pm 0.006}$	$\underline{\mathbf{0.907}}_{\pm 0.000}$
	RF	0.936	$0.648_{\pm 0.036}$	$0.666_{\pm 0.117}$	$0.882_{\pm 0.006}$	$0.897_{\pm 0.011}$	$0.616_{\pm 0.009}$	$\mathbf{0.907}_{\pm 0.007}$	$0.880_{\pm 0.008}$	$0.825_{\pm 0.028}$	$0.873_{\pm 0.006}$	$0.884_{\pm 0.004}$	$\underline{\mathbf{0.925}}_{\pm 0.006}$
	XGB	0.921	$0.731_{\pm 0.020}$	$0.658_{\pm 0.114}$	$0.860_{\pm 0.016}$	$0.885_{\pm 0.010}$	$0.602_{\pm 0.020}$	$\mathbf{0.897}_{\pm 0.010}$	$0.873_{\pm 0.009}$	$0.842_{\pm 0.023}$	$0.862_{\pm 0.005}$	$0.884_{\pm 0.004}$	$\underline{\mathbf{0.917}}_{\pm 0.005}$
wdbc	LN	0.993	$0.929_{\pm 0.053}$	$0.976_{\pm 0.019}$	$0.992_{\pm 0.001}$	$\mathbf{0.993}_{\pm 0.002}$	$0.979_{\pm 0.000}$	$0.988_{\pm 0.003}$	$0.984_{\pm 0.006}$		$0.980_{\pm 0.007}$	$0.979_{\pm 0.007}$	$\underline{\mathbf{0.993}}_{\pm 0.000}$
	RF	0.976	$0.919_{\pm 0.024}$	$0.952_{\pm 0.041}$	$\underline{\mathbf{0.984}}_{\pm 0.004}$	$\mathbf{0.982}_{\pm 0.004}$	$0.976_{\pm 0.003}$	$0.980_{\pm 0.012}$	$0.978_{\pm 0.003}$	-	$0.981_{\pm 0.001}$	$0.975_{\pm 0.002}$	$0.981_{\pm 0.001}$
	XGB	0.990	$0.930_{\pm 0.021}$	$0.949_{\pm 0.035}$	$0.986_{\pm 0.002}$	$\mathbf{0.989}_{\pm 0.001}$	$0.973_{\pm 0.003}$	$0.988_{\pm 0.002}$	$0.982_{\pm 0.004}$		$0.987_{\pm 0.003}$	$0.985_{\pm 0.006}$	$\underline{\mathbf{0.989}}_{\pm 0.001}$

- **Subsample Ratio**: Float in $[0.5, 1.0]$;
- **Subsample Ratio of Columns by Tree**: Float in $[0.5, 1.0]$;
- **L1 Regularization**: Float in $[0.0, 10.0]$.

For all models, numeric values are standardized using standard scaling. Categorical values are one-hot encoded for linear models and label-encoded for random forest and XGBoost. To simplify the experiments, rows with missing values are excluded, as handling missing data is not the focus of this paper.

The reported performance for machine learning utility is measured using the weighted AUC-ROC for classification tasks and the R^2 score for regression tasks, so for all these scores, a larger value indicates a better performance. Each generator is trained and used to generate synthetic data of the same size as the training dataset three times for each dataset, and the summarized results of these runs are reported.

D.5 Fidelity Metrics Implementation

Fidelity metrics are calculated using SDMetrics (sdm, 2023)'s public SDK.

D.6 Distance to Closest Record (DCR) Metrics Implementation

The way the DCR values are reported differ from paper to paper. Also, existing papers often report a value and claim some specific value to be an ideal value, overlooking the fact that better privacy usually sacrifices quality. Thus, in this paper, we instead apply a statistical test on DCR values to *validate* privacy-preserving capability of the models, instead of *evaluating* it, which is likely also more useful in practical privacy-preserving data sharing cases.

We compare the DCR to the real training dataset $\mathbf{X}$ of a hold-out real dataset $\widehat{\mathbf{X}}$ with a synthetic dataset of the same size $\mathbf{X}'$, and privacy-perserving means that DCRs calculated on the latter is not smaller than DCRs calculated on the former. We test this using Mann-Whitney U Test (Mann and Whitney, 1947), with the null hypothesis H_0 being that the distance between $\widehat{\mathbf{X}}$ and $\mathbf{X}$ is greater than or equal to $\mathbf{X}'$ and $\mathbf{X}$, and if the p-value is less than 0.05, $\mathbf{X}'$ is closer to $\mathbf{X}$ than $\widehat{\mathbf{X}}$, implying a risk of privacy leakage.

The data is preprocessed by quantile transformation (at 1000 quantiles, which is the same number of bins for the quantile quantizer in TabTreeFormer) for numeric values and one-hot encoding for categorical values for distance calculation. Cosine distance (calculated using `sklearn`)

Table 11: Performance comparison between different models in terms of fidelity. The closer the values are to 1, the better the fidelity is. The best scores and the second best scores are highlighted in bold with and without underscore, respectively. Equal values in the first 3 digits may be compared by the 4-th or 5-th digits.

Dataset	Metric	CTAB+	TTVAE	TabSyn	FD	GReaT	RTF	TabMT	TabuLa	TTF-S	TTF-L	TTF-NP
adult	Shape	$0.964_{\pm 0.005}$	$0.852_{\pm 0.024}$	$\mathbf{0.977}_{\pm 0.011}$	$\underline{\mathbf{0.986}}_{\pm 0.000}$	$0.929_{\pm 0.000}$	$0.962_{\pm 0.001}$	$0.974_{\pm 0.003}$	$0.975_{\pm 0.000}$	$0.951_{\pm 0.002}$	$0.957_{\pm 0.003}$	$0.921_{\pm 0.003}$
	Trend	$0.915_{\pm 0.004}$	$0.733_{\pm 0.032}$	$0.955_{\pm 0.018}$	$\underline{\mathbf{0.972}}_{\pm 0.000}$	$0.882_{\pm 0.000}$	$0.933_{\pm 0.003}$	$\mathbf{0.957}_{\pm 0.004}$	$0.956_{\pm 0.001}$	$0.887_{\pm 0.003}$	$0.895_{\pm 0.005}$	$0.838_{\pm 0.027}$
bank	Shape	$0.965_{\pm 0.002}$	$0.875_{\pm 0.009}$	$\mathbf{0.983}_{\pm 0.005}$	$0.970_{\pm 0.001}$	$0.915_{\pm 0.000}$	$0.967_{\pm 0.001}$	$0.969_{\pm 0.009}$	$\underline{\mathbf{0.983}}_{\pm 0.000}$	$0.927_{\pm 0.003}$	$0.935_{\pm 0.003}$	$0.932_{\pm 0.001}$
	Trend	$0.884_{\pm 0.003}$	$0.788_{\pm 0.017}$	$\mathbf{0.967}_{\pm 0.008}$	$0.965_{\pm 0.019}$	$0.903_{\pm 0.000}$	$0.955_{\pm 0.001}$	$0.944_{\pm 0.034}$	$\underline{\mathbf{0.971}}_{\pm 0.002}$	$0.944_{\pm 0.006}$	$0.956_{\pm 0.003}$	$0.935_{\pm 0.027}$
boston	Shape	$0.850_{\pm 0.003}$	$0.663_{\pm 0.023}$	$0.893_{\pm 0.003}$	$0.895_{\pm 0.002}$	$\mathbf{0.916}_{\pm 0.000}$	$\underline{\mathbf{0.942}}_{\pm 0.004}$	$0.835_{\pm 0.009}$	$0.861_{\pm 0.030}$	$0.843_{\pm 0.006}$	$0.855_{\pm 0.003}$	$0.863_{\pm 0.000}$
	Trend	$0.828_{\pm 0.011}$	$0.747_{\pm 0.021}$	$0.934_{\pm 0.003}$	$0.942_{\pm 0.001}$	$0.908_{\pm 0.000}$	$0.947_{\pm 0.002}$	$0.935_{\pm 0.014}$	$0.886_{\pm 0.004}$	$0.951_{\pm 0.020}$	$\mathbf{0.973}_{\pm 0.006}$	$\underline{\mathbf{0.992}}_{\pm 0.000}$
breast	Shape	$0.775_{\pm 0.024}$	$0.498_{\pm 0.052}$	$0.813_{\pm 0.005}$	$0.835_{\pm 0.006}$	$0.728_{\pm 0.000}$	$0.753_{\pm 0.014}$	$0.841_{\pm 0.048}$	$0.814_{\pm 0.003}$	$\underline{\mathbf{0.866}}_{\pm 0.022}$	$\mathbf{0.848}_{\pm 0.010}$	$0.758_{\pm 0.116}$
	Trend	$0.616_{\pm 0.063}$	$0.333_{\pm 0.065}$	$0.709_{\pm 0.006}$	$\mathbf{0.766}_{\pm 0.004}$	$0.636_{\pm 0.000}$	$0.668_{\pm 0.010}$	$0.747_{\pm 0.035}$	$\underline{\mathbf{0.767}}_{\pm 0.003}$	$0.735_{\pm 0.025}$	$0.728_{\pm 0.020}$	$0.680_{\pm 0.165}$
credit	Shape	$0.937_{\pm 0.013}$	$0.555_{\pm 0.001}$	$0.932_{\pm 0.007}$	$0.960_{\pm 0.001}$	$0.932_{\pm 0.000}$	$0.944_{\pm 0.003}$	$0.955_{\pm 0.004}$	$\mathbf{0.970}_{\pm 0.002}$	$0.941_{\pm 0.008}$	$0.946_{\pm 0.010}$	$\underline{\mathbf{0.992}}_{\pm 0.008}$
	Trend	$0.862_{\pm 0.030}$	$0.306_{\pm 0.005}$	$0.862_{\pm 0.008}$	$0.912_{\pm 0.009}$	$0.861_{\pm 0.000}$	$0.896_{\pm 0.006}$	$0.911_{\pm 0.004}$	$\mathbf{0.917}_{\pm 0.001}$	$0.880_{\pm 0.012}$	$0.883_{\pm 0.018}$	$\underline{\mathbf{0.958}}_{\pm 0.019}$
diabetes	Shape	$0.923_{\pm 0.006}$	$0.887_{\pm 0.017}$	$\mathbf{0.955}_{\pm 0.007}$	$0.943_{\pm 0.003}$	$0.881_{\pm 0.000}$	$0.943_{\pm 0.004}$	$0.946_{\pm 0.004}$	$\underline{\mathbf{0.975}}_{\pm 0.002}$	$0.928_{\pm 0.002}$	$0.919_{\pm 0.014}$	$0.806_{\pm 0.212}$
	Trend	$0.898_{\pm 0.004}$	$0.888_{\pm 0.012}$	$\mathbf{0.964}_{\pm 0.002}$	$0.957_{\pm 0.006}$	$0.921_{\pm 0.000}$	$0.952_{\pm 0.005}$	$0.952_{\pm 0.003}$	$\underline{\mathbf{0.971}}_{\pm 0.012}$	$0.954_{\pm 0.004}$	$0.957_{\pm 0.000}$	$0.924_{\pm 0.091}$
iris	Shape	$0.822_{\pm 0.022}$	$0.885_{\pm 0.008}$	$0.893_{\pm 0.003}$	$\mathbf{0.901}_{\pm 0.004}$	$0.870_{\pm 0.000}$	$\underline{\mathbf{0.906}}_{\pm 0.017}$	$0.895_{\pm 0.007}$	-	$0.886_{\pm 0.011}$	$0.883_{\pm 0.019}$	$0.839_{\pm 0.029}$
	Trend	$0.590_{\pm 0.019}$	$0.878_{\pm 0.004}$	$0.917_{\pm 0.007}$	$\mathbf{0.920}_{\pm 0.006}$	$0.899_{\pm 0.000}$	$0.896_{\pm 0.016}$	$0.901_{\pm 0.011}$		$0.902_{\pm 0.005}$	$\underline{\mathbf{0.922}}_{\pm 0.018}$	$0.895_{\pm 0.055}$
qsar	Shape	$0.917_{\pm 0.003}$	$0.652_{\pm 0.031}$	$0.930_{\pm 0.005}$	$0.947_{\pm 0.004}$	$0.887_{\pm 0.000}$	$\underline{\mathbf{0.959}}_{\pm 0.003}$	$0.920_{\pm 0.006}$	$0.833_{\pm 0.034}$	$0.914_{\pm 0.004}$	$0.936_{\pm 0.003}$	$\mathbf{0.954}_{\pm 0.001}$
	Trend	$0.863_{\pm 0.007}$	$0.610_{\pm 0.059}$	$0.914_{\pm 0.009}$	$0.898_{\pm 0.011}$	$0.855_{\pm 0.000}$	$0.890_{\pm 0.007}$	$0.923_{\pm 0.004}$	$0.792_{\pm 0.008}$	$0.896_{\pm 0.010}$	$\mathbf{0.935}_{\pm 0.008}$	$\underline{\mathbf{0.956}}_{\pm 0.001}$
wdbc	Shape	$0.861_{\pm 0.005}$	$0.754_{\pm 0.047}$	$0.946_{\pm 0.004}$	$0.946_{\pm 0.004}$	$0.873_{\pm 0.000}$	$0.945_{\pm 0.006}$	$0.937_{\pm 0.008}$	-	$0.938_{\pm 0.007}$	$\mathbf{0.957}_{\pm 0.003}$	$\underline{\mathbf{0.980}}_{\pm 0.004}$
	Trend	$0.866_{\pm 0.007}$	$0.817_{\pm 0.049}$	$\mathbf{0.976}_{\pm 0.002}$	$0.962_{\pm 0.001}$	$0.896_{\pm 0.000}$	$0.954_{\pm 0.001}$	$0.943_{\pm 0.001}$		$0.960_{\pm 0.002}$	$0.966_{\pm 0.006}$	$\underline{\mathbf{0.994}}_{\pm 0.002}$

Table 12: Performance comparison between different models in terms of DCR. p-values less than 0.05 is highlighted in red, indicating a high risk of privacy leakage.

Dataset	CTAB+	TTVAE	TabSyn	FD	GReaT	RTF	TabMT	TabuLa	TTF-S	TTF-L	TTF-NP
adult	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.857_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.998_{\pm 0.002}$	$0.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
bank	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.394_{\pm 0.082}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
boston	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.941_{\pm 0.040}$	$0.996_{\pm 0.005}$	$1.000_{\pm 0.000}$	$0.508_{\pm 0.295}$	$1.000_{\pm 0.000}$	$0.754_{\pm 0.121}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$
breast	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.231_{\pm 0.105}$	$0.003_{\pm 0.001}$	$0.082_{\pm 0.000}$	$0.002_{\pm 0.001}$	$0.259_{\pm 0.064}$	$0.000_{\pm 0.000}$	$0.990_{\pm 0.012}$	$0.996_{\pm 0.005}$	$0.332_{\pm 0.470}$
credit	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.186_{\pm 0.141}$	$0.005_{\pm 0.003}$	$0.883_{\pm 0.000}$	$0.000_{\pm 0.000}$	$0.824_{\pm 0.183}$	$0.000_{\pm 0.000}$	$0.908_{\pm 0.107}$	$0.676_{\pm 0.455}$	$0.000_{\pm 0.000}$
diabetes	$1.000_{\pm 0.000}$	$0.753_{\pm 0.288}$	$0.029_{\pm 0.024}$	$0.000_{\pm 0.000}$	$0.998_{\pm 0.000}$	$0.077_{\pm 0.076}$	$0.876_{\pm 0.149}$	$0.000_{\pm 0.000}$	$0.867_{\pm 0.158}$	$0.603_{\pm 0.310}$	$0.333_{\pm 0.471}$
iris	$1.000_{\pm 0.000}$	$0.382_{\pm 0.229}$	$0.552_{\pm 0.099}$	$0.049_{\pm 0.032}$	$0.764_{\pm 0.000}$	$0.638_{\pm 0.226}$	$0.824_{\pm 0.191}$	-	$0.662_{\pm 0.421}$	$0.214_{\pm 0.022}$	$0.086_{\pm 0.096}$
qsar	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.977_{\pm 0.019}$	$1.000_{\pm 0.000}$	$0.326_{\pm 0.088}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.956_{\pm 0.063}$	$0.000_{\pm 0.000}$
wdbc	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.092_{\pm 0.111}$	$0.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	$0.998_{\pm 0.001}$	$1.000_{\pm 0.000}$	-	$0.967_{\pm 0.021}$	$0.261_{\pm 0.179}$	$0.000_{\pm 0.000}$

is used to evaluate the distance between records. As a reference, we calculate the cosine distances between the real test set and the real train set, obtaining the minimum distance per record for the test set. Similarly, we compute the cosine distances between synthetic data (with the same number of rows as the real test set) and the real train set, and also obtain the minimum distances. These distances are compared with the reference values to assess the similarity and privacy-preserving characteristics of the synthetic data.

E Raw Experimental Results

E.1 Raw Synthetic Data Quality Results

The raw experiment results for MLE is shown in Table 10. Out of the 27 MLE scores, TabTreeFormer-NM is the best in 11, while the best baseline achieves the best in at most 4).

The raw experiment results for fidelity is shown in Table 11.

E.2 Raw Privacy Results

The raw experiment results for DCR is shown in Table 12.

Notes on the No Mask (NM) Setting and Quality-privacy Dilemma. Summarizing the settings of the non-private version mentioned in Appendix A, TTF-NM in the paper means TTF-L model size with 0 masking rate and technically no early stopping. Such a setting makes the model prone to memorizing exact values in the training data, which is verified by our experiment results on DCR. Nevertheless, in a non-private setting, we regard this as permissible because privacy is not a concern.

In fact, for any model, synthetic data quality and its privacy are usually a dilemma, which is also consistent with our experiment results on different baselines. For example, the baseline model with the best MLE result overall is

Table 13: Raw MLE performances as shown in Table 10, but experiments with privacy risks as indicated in Table 12 highlighted and excluded from ranking. TTF-NM is not shown because it is not intended for a balanced objective. Baselines with more than half of the datasets having privacy concerns are removed.

Dataset	ML	real	CTAB+	TTVAE	TabSyn	GReaT	RTF	TabMT	TTF-S	TTF-L
adult	LN	0.914	$0.904_{\pm 0.001}$	$0.875_{\pm 0.010}$	$0.910_{\pm 0.002}$	$\mathbf{0.911}_{\pm 0.000}$	$0.911_{\pm 0.000}$	$0.911_{\pm 0.000}$	$0.906_{\pm 0.001}$	$\underline{\mathbf{0.911}}_{\pm 0.001}$
	RF	0.910	$0.902_{\pm 0.002}$	$0.869_{\pm 0.011}$	$0.905_{\pm 0.003}$	$0.905_{\pm 0.000}$	$0.908_{\pm 0.000}$	$\underline{\mathbf{0.907}}_{\pm 0.002}$	$0.901_{\pm 0.001}$	$\mathbf{0.905}_{\pm 0.000}$
	XGB	0.914	$0.905_{\pm 0.001}$	$0.870_{\pm 0.012}$	$0.910_{\pm 0.003}$	$0.911_{\pm 0.000}$	$0.912_{\pm 0.001}$	$\mathbf{0.911}_{\pm 0.001}$	$0.906_{\pm 0.001}$	$\underline{\mathbf{0.911}}_{\pm 0.001}$
bank	LN	0.907	$0.901_{\pm 0.003}$	$0.885_{\pm 0.004}$	$0.900_{\pm 0.005}$	$\underline{\mathbf{0.907}}_{\pm 0.000}$	$0.904_{\pm 0.000}$	$0.862_{\pm 0.005}$	$0.906_{\pm 0.000}$	$\mathbf{0.906}_{\pm 0.000}$
	RF	0.930	$0.902_{\pm 0.002}$	$0.884_{\pm 0.009}$	$0.907_{\pm 0.006}$	$0.907_{\pm 0.001}$	$\mathbf{0.913}_{\pm 0.001}$	$0.886_{\pm 0.011}$	$0.913_{\pm 0.002}$	$\underline{\mathbf{0.918}}_{\pm 0.001}$
	XGB	0.936	$0.906_{\pm 0.000}$	$0.877_{\pm 0.011}$	$0.913_{\pm 0.007}$	$0.910_{\pm 0.001}$	$0.918_{\pm 0.002}$	$0.892_{\pm 0.007}$	$\mathbf{0.919}_{\pm 0.002}$	$\underline{\mathbf{0.925}}_{\pm 0.003}$
boston	LN	0.590	$0.264_{\pm 0.034}$	$0.000_{\pm 0.079}$	$\mathbf{0.533}_{\pm 0.036}$	$0.397_{\pm 0.000}$	$0.529_{\pm 0.028}$	$0.409_{\pm 0.015}$	$0.525_{\pm 0.035}$	$\underline{\mathbf{0.575}}_{\pm 0.011}$
	RF	0.662	$0.129_{\pm 0.015}$	$0.124_{\pm 0.068}$	$\mathbf{0.658}_{\pm 0.025}$	$0.281_{\pm 0.001}$	$0.584_{\pm 0.006}$	$0.439_{\pm 0.096}$	$0.597_{\pm 0.038}$	$\underline{\mathbf{0.694}}_{\pm 0.041}$
	XGB	0.701	$0.097_{\pm 0.078}$	$0.211_{\pm 0.121}$	$\mathbf{0.647}_{\pm 0.017}$	$0.277_{\pm 0.014}$	$0.578_{\pm 0.056}$	$0.446_{\pm 0.115}$	$0.594_{\pm 0.061}$	$\underline{\mathbf{0.665}}_{\pm 0.026}$
breast	LN	0.985	$0.913_{\pm 0.091}$	$0.963_{\pm 0.006}$	$\underline{\mathbf{0.987}}_{\pm 0.001}$	$0.985_{\pm 0.000}$	$0.987_{\pm 0.002}$	$0.983_{\pm 0.004}$	$0.981_{\pm 0.005}$	$\mathbf{0.987}_{\pm 0.002}$
	RF	0.985	$0.968_{\pm 0.012}$	$0.979_{\pm 0.004}$	$\mathbf{0.981}_{\pm 0.002}$	$0.979_{\pm 0.004}$	$0.980_{\pm 0.000}$	$0.978_{\pm 0.002}$	$0.978_{\pm 0.001}$	$\underline{\mathbf{0.983}}_{\pm 0.002}$
	XGB	0.984	$0.958_{\pm 0.025}$	$0.970_{\pm 0.009}$	$\underline{\mathbf{0.987}}_{\pm 0.001}$	$0.982_{\pm 0.002}$	$0.982_{\pm 0.002}$	$0.982_{\pm 0.002}$	$0.982_{\pm 0.002}$	$\mathbf{0.983}_{\pm 0.003}$
credit	LN	0.836	$0.538_{\pm 0.093}$	$0.500_{\pm 0.000}$	$\mathbf{0.780}_{\pm 0.017}$	$0.665_{\pm 0.000}$	$0.801_{\pm 0.006}$	$0.773_{\pm 0.038}$	$\underline{\mathbf{0.816}}_{\pm 0.020}$	$0.755_{\pm 0.011}$
	RF	0.837	$0.505_{\pm 0.097}$	$0.500_{\pm 0.000}$	$\underline{\mathbf{0.791}}_{\pm 0.007}$	$0.722_{\pm 0.014}$	$0.820_{\pm 0.010}$	$0.763_{\pm 0.019}$	$\mathbf{0.781}_{\pm 0.026}$	$0.741_{\pm 0.027}$
	XGB	0.844	$0.509_{\pm 0.088}$	$0.500_{\pm 0.000}$	$\mathbf{0.795}_{\pm 0.014}$	$0.739_{\pm 0.006}$	$0.797_{\pm 0.039}$	$0.761_{\pm 0.030}$	$\underline{\mathbf{0.796}}_{\pm 0.011}$	$0.764_{\pm 0.014}$
diabetes	LN	0.884	$0.817_{\pm 0.028}$	$0.880_{\pm 0.003}$	$0.883_{\pm 0.006}$	$0.858_{\pm 0.000}$	$0.878_{\pm 0.003}$	$0.855_{\pm 0.014}$	$\underline{\mathbf{0.890}}_{\pm 0.010}$	$\mathbf{0.886}_{\pm 0.001}$
	RF	0.869	$0.802_{\pm 0.014}$	$0.849_{\pm 0.004}$	$0.844_{\pm 0.012}$	$0.824_{\pm 0.012}$	$0.841_{\pm 0.010}$	$0.845_{\pm 0.009}$	$\mathbf{0.855}_{\pm 0.014}$	$\underline{\mathbf{0.859}}_{\pm 0.010}$
	XGB	0.867	$0.811_{\pm 0.005}$	$\mathbf{0.851}_{\pm 0.001}$	$0.847_{\pm 0.009}$	$0.831_{\pm 0.003}$	$0.849_{\pm 0.010}$	$0.840_{\pm 0.009}$	$\underline{\mathbf{0.852}}_{\pm 0.015}$	$0.848_{\pm 0.017}$
iris	LN	1.000	$0.272_{\pm 0.068}$	$0.996_{\pm 0.006}$	$\mathbf{0.999}_{\pm 0.002}$	$0.985_{\pm 0.000}$	$0.983_{\pm 0.015}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$0.971_{\pm 0.018}$	$0.997_{\pm 0.005}$
	RF	1.000	$0.396_{\pm 0.099}$	$0.992_{\pm 0.012}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$0.993_{\pm 0.005}$	$0.993_{\pm 0.005}$	$0.999_{\pm 0.002}$	$0.988_{\pm 0.017}$
	XGB	1.000	$0.211_{\pm 0.042}$	$0.999_{\pm 0.002}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$0.998_{\pm 0.003}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$	$\underline{\mathbf{1.000}}_{\pm 0.000}$
qsar	LN	0.906	$0.716_{\pm 0.085}$	$0.669_{\pm 0.120}$	$0.867_{\pm 0.012}$	$0.673_{\pm 0.000}$	$\mathbf{0.880}_{\pm 0.003}$	$\underline{\mathbf{0.887}}_{\pm 0.005}$	$0.874_{\pm 0.013}$	$0.876_{\pm 0.006}$
	RF	0.936	$0.648_{\pm 0.036}$	$0.666_{\pm 0.117}$	$0.882_{\pm 0.006}$	$0.616_{\pm 0.009}$	$\underline{\mathbf{0.907}}_{\pm 0.007}$	$0.880_{\pm 0.008}$	$0.873_{\pm 0.006}$	$\mathbf{0.884}_{\pm 0.004}$
	XGB	0.921	$0.731_{\pm 0.020}$	$0.658_{\pm 0.114}$	$0.860_{\pm 0.016}$	$0.602_{\pm 0.020}$	$\underline{\mathbf{0.897}}_{\pm 0.010}$	$0.873_{\pm 0.009}$	$0.862_{\pm 0.005}$	$\mathbf{0.884}_{\pm 0.004}$
wdbc	LN	0.993	$0.929_{\pm 0.053}$	$0.976_{\pm 0.019}$	$\underline{\mathbf{0.992}}_{\pm 0.001}$	$0.979_{\pm 0.000}$	$\mathbf{0.988}_{\pm 0.003}$	$0.984_{\pm 0.006}$	$0.980_{\pm 0.007}$	$0.979_{\pm 0.007}$
	RF	0.976	$0.919_{\pm 0.024}$	$0.952_{\pm 0.041}$	$\underline{\mathbf{0.984}}_{\pm 0.004}$	$0.976_{\pm 0.003}$	$0.980_{\pm 0.012}$	$0.978_{\pm 0.003}$	$\mathbf{0.981}_{\pm 0.001}$	$0.975_{\pm 0.002}$
	XGB	0.990	$0.930_{\pm 0.021}$	$0.949_{\pm 0.035}$	$0.986_{\pm 0.002}$	$0.973_{\pm 0.003}$	$\underline{\mathbf{0.988}}_{\pm 0.002}$	$0.982_{\pm 0.004}$	$\mathbf{0.987}_{\pm 0.003}$	$0.985_{\pm 0.006}$

TabuLa (Zhao et al., 2023), but it has a privacy issue in *all* datasets. In particular, we found that TabuLa has a DCR of 0 on more than 50% of the rows in many datasets, and a DCR of smaller than 1×10^{-6} on more than 80% of the rows. This indicates a severe memorization issue. In comparison, the worst-performing baseline model we tested, CTAB-GAN+ (Zhao et al., 2024), does not have any issue with privacy according to the DCR scores.

Balance of Quality and Utility. Given the dilemma between quality and utility, the general objective of a tabular data generator should be a good balance between quality and privacy, instead of optimizing both simultaneously. Table 13 shows the MLE scores with privacy warnings. TabTreeFormer demonstrates a good balance.

E.3 Computation Time

The raw computation time of training and generation is shown in Table 14. TabTreeFormer trains two separate models for cross-validation, and training stops mainly by the validation loss, and occasionally by a maximum number of steps, so the relative training time compared to other auto-regressive transformer baselines is higher, but when trained on large datasets, TabTreeFormer shows its efficiency advantage more obviously.

F Limitations and Future Work

Limited capability by transformer backbones. Although this paper demonstrates a strong capability of TabTreeFormer and a potential of comparable performance with a much smaller version, it still inherits the problems of the backbone transformer. In particular, TabTreeFormer's tabular data representation effectively reduces the number of tokens to one to two times the number of columns, significantly smaller than the 5-10 times of some baseline auto-regressive transformers, but is still linear to the number of columns. If the chosen transformer backbone

Table 14: Comparison between different models in terms of time taken for training and generation. The values are in unit of seconds.

Dataset	Metric	CTAB+	TTVAE	TabSyn	FD	GReaT	RTF	TabMT	TabuLa	TTF-S	TTF-L
adult	Train	$787.379_{\pm 3.334}$	$194.718_{\pm 0.085}$	$1199.823_{\pm 146.744}$	$3857.698_{\pm 10.002}$	$10397.040_{\pm 9.203}$	$1875.904_{\pm 34.663}$	$6029.515_{\pm 3.139}$	$2859.160_{\pm 6.858}$	$1096.467_{\pm 143.847}$	$2503.830_{\pm 564.922}$
	Generate	$0.429_{\pm 0.001}$	$8.546_{\pm 0.029}$	$2.409_{\pm 0.010}$	$48.730_{\pm 13.758}$	$138.023_{\pm 0.205}$	$89.466_{\pm 4.301}$	$416.206_{\pm 1.612}$	$249.319_{\pm 0.485}$	$41.482_{\pm 8.907}$	$107.193_{\pm 24.859}$
bank	Train	$737.314_{\pm 0.138}$	$192.617_{\pm 1.959}$	$1865.473_{\pm 418.665}$	$4172.922_{\pm 3.229}$	$10295.769_{\pm 21.720}$	$835.167_{\pm 1.619}$	$7032.875_{\pm 4.217}$	$3079.641_{\pm 2.237}$	$1115.787_{\pm 108.995}$	$2551.349_{\pm 724.774}$
	Generate	$1.126_{\pm 0.006}$	$7.341_{\pm 0.015}$	$3.009_{\pm 0.063}$	$31.609_{\pm 0.373}$	$149.872_{\pm 0.104}$	$106.036_{\pm 0.080}$	$333.882_{\pm 118.034}$	$234.806_{\pm 0.044}$	$24.972_{\pm 3.040}$	$58.530_{\pm 16.181}$
boston	Train	$8.396_{\pm 0.048}$	$4.664_{\pm 0.129}$	$862.625_{\pm 48.241}$	$47.040_{\pm 1.544}$	$121.619_{\pm 0.090}$	$49.693_{\pm 3.175}$	$225.624_{\pm 0.310}$	$47.811_{\pm 0.079}$	$67.370_{\pm 43.192}$	$229.997_{\pm 98.326}$
	Generate	$0.083_{\pm 0.001}$	$0.172_{\pm 0.029}$	$0.156_{\pm 0.003}$	$0.447_{\pm 0.007}$	$2.422_{\pm 0.003}$	$3.059_{\pm 0.003}$	$2.524_{\pm 0.001}$	$3.012_{\pm 0.013}$	$0.533_{\pm 0.028}$	$1.566_{\pm 0.401}$
breast	Train	$8.479_{\pm 0.026}$	$6.355_{\pm 0.592}$	$1037.126_{\pm 281.166}$	$32.364_{\pm 0.506}$	$153.352_{\pm 0.134}$	$25.537_{\pm 1.419}$	$214.935_{\pm 0.461}$	$33.100_{\pm 0.059}$	$122.137_{\pm 6.839}$	$344.247_{\pm 59.644}$
	Generate	$0.056_{\pm 0.008}$	$0.144_{\pm 0.032}$	$0.249_{\pm 0.003}$	$0.771_{\pm 0.005}$	$3.434_{\pm 0.016}$	$0.759_{\pm 0.003}$	$1.653_{\pm 0.013}$	$6.652_{\pm 0.011}$	$0.620_{\pm 0.033}$	$1.463_{\pm 0.208}$
credit	Train	$8.400_{\pm 0.035}$	$12.288_{\pm 2.809}$	$726.006_{\pm 67.662}$	$185.868_{\pm 5.293}$	$277.555_{\pm 0.388}$	$63.648_{\pm 0.758}$	$703.238_{\pm 1.755}$	$78.538_{\pm 0.338}$	$179.304_{\pm 56.926}$	$458.389_{\pm 89.990}$
	Generate	$0.043_{\pm 0.000}$	$0.211_{\pm 0.115}$	$0.167_{\pm 0.003}$	$1.127_{\pm 0.010}$	$5.845_{\pm 0.008}$	$2.407_{\pm 0.171}$	$5.434_{\pm 0.013}$	$4.868_{\pm 0.014}$	$0.604_{\pm 0.123}$	$1.558_{\pm 0.227}$
diabetes	Train	$8.524_{\pm 0.407}$	$6.673_{\pm 0.480}$	$758.140_{\pm 41.333}$	$30.324_{\pm 2.067}$	$130.793_{\pm 0.077}$	$37.372_{\pm 3.118}$	$220.147_{\pm 0.667}$	$40.229_{\pm 0.050}$	$70.158_{\pm 26.658}$	$242.176_{\pm 48.014}$
	Generate	$0.047_{\pm 0.001}$	$0.064_{\pm 0.001}$	$0.139_{\pm 0.000}$	$0.383_{\pm 0.003}$	$1.496_{\pm 0.008}$	$1.504_{\pm 0.007}$	$0.786_{\pm 0.000}$	$3.994_{\pm 0.007}$	$0.343_{\pm 0.062}$	$1.169_{\pm 0.364}$
iris	Train	$7.267_{\pm 0.021}$	$2.489_{\pm 0.005}$	$620.864_{\pm 35.603}$	$5.055_{\pm 1.403}$	$24.975_{\pm 0.018}$	$15.369_{\pm 1.242}$	$32.846_{\pm 0.011}$	-	$86.143_{\pm 41.970}$	$126.315_{\pm 29.440}$
	Generate	$0.033_{\pm 0.006}$	$0.025_{\pm 0.005}$	$0.098_{\pm 0.002}$	$0.095_{\pm 0.002}$	$0.360_{\pm 0.015}$	$0.225_{\pm 0.001}$	$0.122_{\pm 0.031}$		$0.209_{\pm 0.065}$	$0.483_{\pm 0.078}$
qsar	Train	$18.389_{\pm 0.217}$	$13.144_{\pm 1.576}$	$694.890_{\pm 76.489}$	$664.578_{\pm 3.603}$	$566.658_{\pm 23.451}$	$152.950_{\pm 0.819}$	$1723.824_{\pm 2.596}$	$188.475_{\pm 0.374}$	$81.433_{\pm 14.458}$	$334.639_{\pm 74.700}$
	Generate	$0.206_{\pm 0.006}$	$0.432_{\pm 0.006}$	$0.336_{\pm 0.005}$	$2.608_{\pm 0.007}$	$46.712_{\pm 3.062}$	$10.997_{\pm 0.072}$	$23.723_{\pm 0.025}$	$20.261_{\pm 5.972}$	$1.270_{\pm 0.094}$	$4.118_{\pm 0.606}$
wdbc	Train	$10.795_{\pm 0.089}$	$8.168_{\pm 0.171}$	$631.546_{\pm 68.070}$	$306.120_{\pm 3.189}$	$265.340_{\pm 2.411}$	$183.908_{\pm 1.047}$	$643.120_{\pm 1.028}$	-	$31.884_{\pm 4.836}$	$169.320_{\pm 54.606}$
	Generate	$0.159_{\pm 0.001}$	$0.254_{\pm 0.001}$	$0.276_{\pm 0.005}$	$1.267_{\pm 0.023}$	$65.821_{\pm 1.106}$	$9.936_{\pm 0.017}$	$13.893_{\pm 0.068}$		$0.585_{\pm 0.010}$	$1.381_{\pm 0.278}$

has limited performance or impractical memory requirement for the sequence length induced by a dataset with a tremendous number of columns, the performance of TabTreeFormer is subject to the capability of the transformer backbone. Fortunately, most transformers are believed to function where with a few thousand of tokens in a sequence, endorsing the encoding of tabular datasets of the number of columns up to thousands, which covers most practical use cases. Moreover, the design of TabTreeFormer makes the transformer backbone flexibly configurable, and any modern or future more powerful versions of transformers can be substituted in.

Optimizing tree-based model. TabTreeFormer uses LightGBM fitted on the target column to introduce tabular inductive biases from tree-based models, While this is effective and easy to implement, modified tree-based models catered for guiding the generation may further improve the performance, such as using trees from a generative tree-based model (e.g. ARF), and selecting a better target column using some heuristics.

Smoothing quantized values. The dual-quantization tokenization may be further optimized with some sampling around the quantile values if use case requires smoother values in continuous values. Adapted ordinal embeddings with continuous raw values as additional input can also be applied. In this paper, we want to focus on showing the effectiveness of quantization and methods to handle quantized values. We leave the integration of raw continuous values on top of these methods for future works, as performance without raw continuous values is readily outstanding. Moreover, using quantized values only makes it much easier to change the backbone transformer.

Masked vs. Auto-regressive transformers. TabMT has demonstrated the effectiveness of a masked transformer with iterative decoding for tabular data generation, but TabMT does not provide ablation study result on masked vs. auto-regressive transformers. Thus, this approach may be used in place of a causal language model to further improve synthetic data quality. Further exploration could be interesting. In this paper, we want to focus on the most intuitive and simple usage of generative transformers, namely, auto-regressive generation, as the performance of this simple case is readily outstanding. Moreover, using the standard auto-regressive setting makes it much easier to integrate with other training optimizations available on open-source seq2seq trainers (e.g., Hugging Face).

G Broader Impact

Extension to general tabular tasks. The idea of introducing tree-based models to transformers on tabular data may not be limited to generation tasks, but also applies to tasks like classification and regression. We envision tabular models designed for other tasks inspired by TabTreeFormer's design of the integration of a tree-based model to transformer would improve its performance by taking the advantage of both.

General ordinal token space learning. The proposed methods to learn the ordinal token space, including the embedding and loss, can be applied to any other task involving ordinal tokens, especially when the ordinal tokens' relation is absolute and monotonic instead of relative and non-periodic.

Checklist

- Includes a conceptual outline and/or pseudocode description of AI methods introduced. **Yes**.
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results. **Yes**.
- Provides well marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper. **Yes**.
- Does this paper make theoretical contributions? **Yes**.
  - All assumptions and restrictions are stated clearly and formally. **Yes**.
  - All novel claims are stated formally (e.g., in theorem statements). **Yes**.
  - Proofs of all novel claims are included. **Partial, trivial proofs are omitted.**
  - Proof sketches or intuitions are given for complex and/or novel results. **Yes**.
  - Appropriate citations to theoretical tools used are given. **Yes**.
  - All theoretical claims are demonstrated empirically to hold. **Yes**.
  - All experimental code used to eliminate or disprove claims is included. **Yes**.
- Does this paper rely on one or more datasets? **Yes**.
  - A motivation is given for why the experiments are conducted on the selected datasets. **Yes**.
  - All novel datasets introduced in this paper are included in a data appendix. **Yes**.
  - All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. **Yes**.
  - All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. **Yes**.
  - All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. **Yes**.
  - All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. **NA**.
- Does this paper include computational experiments? **Yes**.
  - This paper states the number and range of values tried per (hyper-)parameter during development of the paper, along with the criterion used for selecting the final parameter setting. **Yes**.
  - Any code required for pre-processing data is included in the appendix. **Yes**.
  - All source code required for conducting and analyzing the experiments is included in a code appendix. **Yes**.
  - All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. **Yes**.
  - All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from. **Yes**.
  - If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. **Yes**.
  - This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. **Yes**.
  - This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. **Yes**.
  - This paper states the number of algorithm runs used to compute each reported result. **Yes**.
  - Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. **Yes**.
  - The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). **Yes**.
  - This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. **Yes**.