TABFAIRGDT: A Fast Fair Tabular Data Generator using Autoregressive Decision Trees

Emmanouil Panagiotou[†§*], Benoît Ronval[‡*], Arjun Roy[†§],
Ludwig Bothmann[¶], Bernd Bischl[¶], Siegfried Nijssen[‡∥], and Eirini Ntoutsi[§]

[†] Freie Universität Berlin, Berlin, Germany
[‡] ICTEAM, UCLouvain, Louvain-la-Neuve, Belgium
[§] Research Institute CODE, Universität der Bundeswehr München, Munich, Germany
[¶] Department of Statistics, Munich Center for Machine Learning (MCML), LMU Munich, Munich, Germany
[∥] DTAI, KU Leuven, Leuven, Belgium
[*]Authors contributed equally to this research
Correspondence: emmanouil.panagiotou@fu-berlin.de, benoit.ronval@uclouvain.be

Abstract—Ensuring fairness in machine learning remains a significant challenge, as models often inherit biases from their training data. Generative models have recently emerged as a promising approach to mitigate bias at the data level while preserving utility. However, many rely on deep architectures, despite evidence that simpler models can be highly effective for tabular data. In this work, we introduce TABFAIRGDT, a novel method for generating fair synthetic tabular data using autoregressive decision trees. To enforce fairness, we propose a soft leaf resampling technique that adjusts decision tree outputs to reduce bias while preserving predictive performance. Our approach is non-parametric, effectively capturing complex relationships between mixed feature types, without relying on assumptions about the underlying data distributions. We evaluate TABFAIRGDT on benchmark fairness datasets and demonstrate that it outperforms state-of-the-art (SOTA) deep generative models, achieving better fairness-utility trade-off for downstream tasks, as well as higher synthetic data quality. Moreover, our method is lightweight, highly efficient, and CPU-compatible, requiring no data pre-processing. Remarkably, TABFAIRGDT achieves a 72% average speedup over the fastest SOTA baseline across various dataset sizes, and can generate fair synthetic data for medium-sized datasets (10 features, 10K samples) in just one second on a standard CPU, making it an ideal solution for real-world fairness-sensitive applications.

Index Terms—Fair Synthetic Data, Generative Modeling, Autoregressive Generation, Decision Trees, Non-parametric Models.

I. INTRODUCTION

Machine learning (ML) models are increasingly deployed in critical sectors like finance, healthcare, and education. However, these models often showcase bias towards individuals or specific demographic groups based on sensitive attributes such as race, gender, age, or socio-economic status. Such biases can arise from various sources [1], including historical discrimination, data collection methodologies [2], and *algorithmic biases*, where design choices within the algorithm, such as the optimization function, introduce bias despite unbiased input data. One potential discriminatory outcome is *group (un-) fairness*, where certain demographic groups receive biased predictions. This occurs when the model systematically favors or disadvantages certain groups based on sensitive attributes. As a result, ensuring fairness in ML models is essential, and a substantial body of literature has been devoted to addressing this issue, ranging from basic philosophical frameworks [3] to technical solutions, typically categorized into pre-, in-, and, post-processing approaches, proposing interventions for fairness at the data-, algorithm-, or model-level, respectively [4]. We focus on approaches that address fairness at the data level, particularly for tabular data, which is the most prevalent data type in real-world applications [5].

Fairness at the data level involves modifying the training data "towards fairness" before feeding them to an ML algorithm. The notion of fair data is defined in various ways in the literature, depending on the specific application or context. Most often, it is framed as achieving statistical parity with respect to a known sensitive attribute and a specific target variable [6], [7]. However, alternative definitions exist, such as, counterfactual fairness [8], ϵ-fairness, when fairness is pursued without a specific downstream target [9], or even in situations where sensitive attributes are not observed [3]. To achieve fair data, various pre-processing approaches have been used, by transforming, reweighting, massaging class labels [10], or augmenting data to mitigate bias, ensuring that sensitive attributes do not unfairly influence predictions. While traditional augmentation techniques, such as oversampling [11]–[13] or reweighting, adjust the existing dataset, generative models [6], [14] offer a more advanced alternative by synthesizing entirely new data that adheres to fairness constraints, preserving statistical properties while reducing bias. Most such methods rely on deep architectures, often leading to longer computational times, even when using GPUs. However, simpler models have proven to remain effective for tabular data generation, while requiring fewer resources in terms of both time and computational power [12], [13].

In this work, we focus on *generating* fair synthetic data, following a common assumption in synthetic data generation for fairness [6], [7], [11], [15], where both the sensitive attribute and the target attribute are observed. The goal is to

generate fair data that (i) preserves the statistical properties of the original data (ii) eliminates discrimination, and (iii) maintains utility on the downstream task. To achieve this, we propose TABFAIRGDT (*Tabular Fair Generative Decision Trees*). Our approach follows an autoregressive framework, generating attributes sequentially using decision trees (DTs). To ensure fairness, we incorporate a *fair leaf resampling* algorithm in the final step when generating the target attribute. This intervention enforces statistical independence between the target and the sensitive attribute in the generated data while preserving the underlying attribute distributions. TABFAIRGDT is a non-parametric approach, making no assumptions about the distributions of attributes, and needs no pre-processing, making it ideal for mixed tabular data, which are common in this domain [16]. Our method is simple yet highly effective, consistently maintaining utility, measured by downstream ROC AUC performance, and data quality, while significantly improving fairness, assessed by downstream statistical parity. Experimental evaluation on fairness benchmark datasets confirms that TABFAIRGDT achieves superior results compared to existing approaches. Furthermore, it is efficient due to the use of DTs, generating fair synthetic data for medium-sized datasets in just one second on a standard computer. Our approach is also lightweight and CPU-compatible, eliminating the need for specialized hardware. All these advantages make TABFAIRGDT ideal for real-world applications. We provide a codebase* to reproduce all experiments.

In summary, the contributions of this work are the following:

- We introduce **TABFAIRGDT**, a novel fair synthetic data generation method using an autoregressive decision tree (DT) framework. It is *efficient* and *non-parametric*, requiring no pre-processing or specialized hardware.
- We propose a *fair leaf resampling* algorithm to enforce fairness while preserving data quality and utility.
- TABFAIRGDT outperforms existing approaches in *fairness-utility tradeoff*, improving fairness by $\sim 50\%$ on average while preserving high predictive score.
- Our method avoids generating sensitive attribute-specific out-of-distribution samples, making it suitable for real-world critical applications like healthcare.

II. RELATED WORK

Recent years have seen an increase in contributions tackling fairness issues in machine learning. Metrics to measure (un-)fairness can be divided [17] into group and individual fairness. Methods to address (un-)fairness, on the other hand, can be divided into pre-processing, in-processing, and post-processing [4]. Our method falls into the category of pre-processing methods, generating fair synthetic data that subsequently can be used for creating models adhering to group fairness. Although there are multiple ways to evaluate fairness, we follow related work for fair synthetic data generation, primarily reporting on statistical parity [6], [7], [11], [15], measuring the difference in positive outcome rate between the sensitive attribute groups

(c.f. Section IV-D for a formal definition). Hereafter, we analyze works relevant to our method, focusing on generative models for tabular data and their fairness-aware variations. Furthermore, since our method is tree-based, we provide an overview of fair DTs.

Generative Models for Tabular Data: Generative models have been used for different data types, from images to text and audio, including tabular data. One family of methods adapts deep learning architectures from other domains for tabular data. For example, CTGAN and TVAE [18] use Generative Adversarial Networks (GANs) and Variational Auto-Encoders to generate tabular data. More recently, diffusion-based models have outperformed them by reversing a gradual noising process to reconstruct tabular data, either directly in input space [19] or via latent diffusion [20]. Other categories of methods exist that do not specifically rely on deep learning architectures. For instance, by approximating the joint distribution of tabular data using n-way marginals [21], estimating densities through random forests [22], or employing distance-based interpolation [23]. In [24], a generative model utilizing Classification and Regression Trees (CART) is introduced for sequential column-wise data generation. Our method builds upon this approach by integrating fairness constraints.

Fair Generative Models for Tabular Data: Methods that introduce fair generative models are closely related to our work. Most approaches follow a two-step process: first, training a deep generative model, such as TabFairGAN [6] and CuTS [15], and then fine-tuning it for fairness using a regularized loss function. Similarly, TabularARGN [7], [14], commercially developed by Mostly AI†, uses a deep autoregressive model and optimizes for fairness only when generating the target variable. PreFair [25] utilizes marginals and a causal model to achieve fairness. Finally, some methods focus on fair data augmentation for the underrepresented groups [12], [13], [26]. Most of these approaches build upon SMOTE [23], such as FSMOTE [11]. These fair data generators serve as baselines in our evaluation, and are described in detail in Section IV-C.

Fair Decision Trees: Our work closely relates to fairness-aware DTs, which incorporate fairness constraints into the DT induction process. Seminal work by [27] introduced two strategies: (i) fair splitting criteria, which modify the tree construction process to enforce fairness during split selection, and (ii) leaf relabeling, which adjusts leaf labels post-construction to enhance fairness. Their study concluded that leaf relabeling was the most effective of the two strategies. Subsequently, [28] integrated a fair splitting criterion within Hoeffding trees to ensure fairness in streaming data. More recently, [29] used a fair splitting criterion to facilitate fairness-aware representation learning. Our approach draws inspiration specifically from the leaf-relabeling strategy and employs a fair leaf-relabeling method to generate fair synthetic data. Although we also experimented with the fair splitting criterion, we found that it introduced substantial structural changes to the DT, compromising data utility.

*TABFAIRGDT code: github.com/Panagiotou/TABFAIRGDT

†https://mostly.ai/

III. TABFAIRGDT: FAST FAIR TABULAR DATA GENERATION USING AUTOREGRESSIVE DECISION TREES

An overview of our approach is shown in Figure 1. Our method consists of two main components: an autoregressive tree-based generation step (Section III-B), and a fairness-utility tradeoff-aware generation step (Section III-C). Before presenting the components in detail, we first present the problem formulation in Section III-A.

A. Problem Formulation

Let D be a dataset defined as $D \subseteq X \times S \times Y$ where $X = X_0 \times X_1 \times \cdots \times X_n$ represents n numerical or categorical features, S is a sensitive attribute, and Y is the target attribute. Following common practice in fairness-aware learning methods and datasets [16], we assume that both $S \in \{0, 1\}$ and $Y \in \{0, 1\}$ are binary variables. For example, S could represent $sex \in \{male, female\}^*$, and Y could represent $income \in \{low, high\}$.

The goal of fair generative AI, as defined in the relevant literature [6], [7], [15], is to learn a generative model based on D, capable of generating a synthetic dataset $\hat{D} \subseteq \hat{X} \times \hat{S} \times \hat{Y}$ that satisfies the following desiderata/requirements:

R1 - Data quality: the generated data should approximate the original data, i.e., $\hat{X} \sim X$ and $\hat{S} \sim S$.

R2 - Fairness control: the target should be statistically independent of the sensitive attribute, i.e., $\hat{Y} \perp \hat{S}$.

R3 - Utility preservation: the utility of the data for the downstream task should be preserved, i.e., $\hat{Y} \sim P(Y|X)$.

To achieve R1 (data quality), features are generated in an autoregressive fashion, where each feature is conditioned on the previously generated features (Section III-B). R2 (fairness control) and R3 (utility preservation) are often in conflict [30], as statistical dependencies frequently exist between the sensitive attribute S and the target variable Y in the original dataset. Enforcing fairness (R2) by reducing or removing these dependencies can limit the model's ability to preserve predictive performance (R3), thereby leading to an inherent trade-off between fairness and utility. We adress this trade-off through a fairness-aware target generation (Section III-C).

B. Autoregressive Tree-based Generation

The autoregressive synthetic data generation process can be formulated as sampling from a sequence of estimated conditional distributions, where each variable's synthetic value depends on the previously generated synthetic values. Given the dataset features $X_0, X_1, \cdots, X_n$, we formally define the process as follows ([14], [24], [31]):

$$\hat{X}_0 \sim P_0(X_0)$$
$$\hat{X}_1 \sim P_1(X_1 \mid X_0)$$
$$\hat{X}_2 \sim P_2(X_2 \mid X_1, X_0)$$
$$\vdots$$
$$\hat{X}_n \sim P_n(X_n \mid X_{<n})$$

*Sex is considered binary in the datasets, not as a reflection of broader societal views.

Algorithm 1 Decision Tree-based Synthetic Data Generation

Input $X_{<j}$, real values for previous features; X_j real values for current feature j; $\hat{X}_{<j}$ synthetic values for previous features.

Output $\hat{X}_j$ synthetic values for feature j.

---

i) Fitting:
1: Train decision tree $Tree(X_{<j}) \to X_j$
2: Initialize probability dictionary $\mathcal{P} \leftarrow \{\}$
3: **for** each leaf ℓ in $Tree$ **do**
4: $\quad p_\ell(X_j | X_{<j} \in \ell) := \frac{\text{count}(X_j = x_j \in \ell)}{\text{count}(X_j \in \ell)} \, \forall \, x_j \in X_j$ in ℓ
5: $\quad \mathcal{P}[\ell] \leftarrow p_\ell(X_j | X_{<j} \in \ell) \quad \triangleright$ Store out probs per leaf
6: **end for**

---

ii) Sampling:
7: Initialize dictionary $\mathcal{ID} \leftarrow \{\ell \to [\,] \mid \ell \in Tree\}$
8: **for** each index i in $\{1, 2, \ldots, |\hat{X}_{<j}|\}$ **do** $\triangleright$ Loop over all samples
9: $\quad \hat{x} \leftarrow \hat{X}_{<j}[i]$
10: $\quad$ Find leaf $\ell = Tree(\hat{x})$
11: $\quad \mathcal{ID}[\ell] \leftarrow \mathcal{ID}[\ell] \cup \{i\} \triangleright$ Store indices of $\hat{X}_{<j}$ per leaf
12: **end for**
13: $\hat{X}_j \leftarrow [\,]$
14: **for** each leaf ℓ in $\mathcal{ID}$ **do**
15: $\quad idxs \leftarrow \mathcal{ID}[\ell] \quad \triangleright$ Get all indices of $\hat{X}_{<j}$ in each leaf
16: $\quad n \leftarrow \text{length}(idxs)$
17: $\quad \hat{X}_j[idxs] \leftarrow \text{sample}(\mathcal{P}[\ell], n) \quad \triangleright$ Draw n synthetic samples from p_ℓ
18: **end for**
19: **Return** $\hat{X}_j$

The autoregressive process generates each feature $X_j, j \in \{0, \ldots, n\}$ sequentially, conditioned on prior synthetic values. Similarly, the sensitive attribute S is generated, given all other columns X, i.e. $\hat{S} \sim P_S(S \mid X)$. To model the conditional probabilities P_j for a feature X_j we use DTs due to their efficiency, flexibility, and ability to handle diverse data types ([24]). The iterative per-feature generation process consists of two phases: fitting and sampling (c.f. Algorithm 1):

(i) **Fitting phase**: For a feature X_j, a DT is trained to map $X_{<j}$ to X_j, storing the probability output distribution for each leaf in a dictionary $\mathcal{P}$, effectively estimating $P_j(X_j \mid X_{<j})$ (c.f. Algorithm 1 line 5).

(ii) **Sampling phase**: Synthetic data of the previous columns $\hat{X}_{<j}$ is passed through the tree, and samples are drawn from $\mathcal{P}$ based on leaf assignments, to generate the next synthetic column $\hat{X}_j$ (c.f. Algorithm 1 line 17).

The same fitting-sampling approach is used to generate all columns $\hat{X}_j$, except for the first feature, $\hat{X}_0$, which is sampled at random from X_0, since there are no previous synthetic values to generate from. This autoregressive generation process ensures the quality of data generation (R1).

Discussion on the autoregressive generation process: TABFAIRGDT employs DTs at its core due to their simplicity, efficiency, and non-parametric nature, i.e. making no assump-
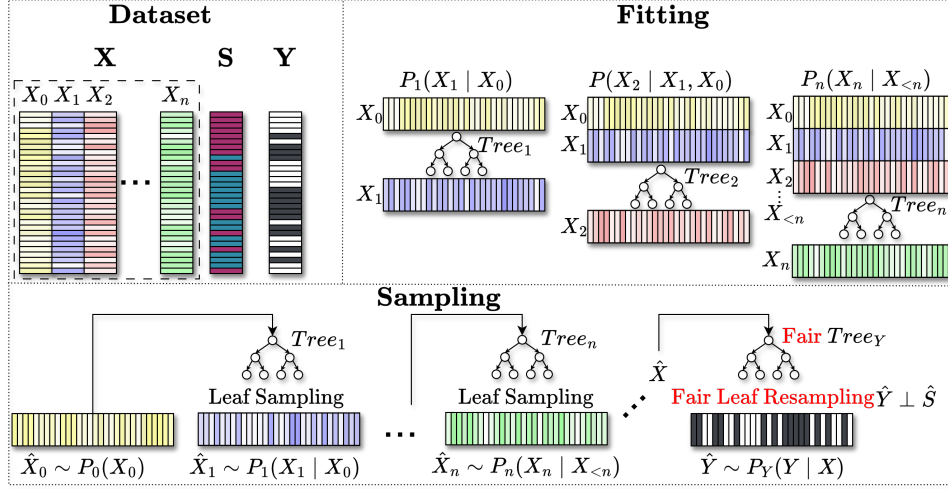
Fig. 1: Overview of the synthetic data generation in TABFAIRGDT.

tion on the underlying data distribution (as previously explored in [24]). In the context of fair data generation, simpler models have been shown to outperform more complex, generative approaches [12], [13]. Our generation process proceeds in the original order of features as present in the dataset. As shown in Section V-C, the feature generation order does *not* significantly impact the results. Our autoregressive approach assumes correlations between certain features, as observed in real datasets. If all features are fully independent, the applicability of our approach would be limited, although such datasets are rarely useful in practice.

C. Fairness-Aware Target Generation

In the previous section, we described the autoregressive generation process using DTs, highlighting its flexibility, which allows to generate the sensitive attribute $\hat{S}$ after all other attributes have been produced. Moreover, it allows enforcing fairness constraints exclusively at the final step, when generating the target variable, ensuring fairness control (R2) and utility preservation (R3). Enforcing fairness earlier in the generation process may lead to out-of-distribution samples as observed in our experiments (c.f. Section V-B). Such deviations directly reduce data quality (R1) and can be particularly problematic in high-stakes domains such as healthcare, where data fidelity is critical. Imposing fairness constraints only in the final step has also been supported by recent literature, including [7], [14], which apply such constraints exclusively during target generation in deep autoregressive frameworks.

Our goal is to generate fair data by ensuring statistical parity in downstream tasks. In this work, we focus on statistical parity as our fairness criterion, acknowledging that fairness is inherently domain-specific and that the appropriate fairness measure should be selected accordingly[2]. Specifically, we

generate the synthetic target variable $\hat{Y}$ so that a model's predictions Y_{pred}, satisfy statistical parity:

$$P(Y_{pred} = 1|S = 0) = P(Y_{pred} = 1|S = 1)$$

To achieve this, we build upon the fair leaf relabeling approach [27], a post-processing bias mitigation technique that modifies a DT by performing hard relabeling of its leaf nodes to reduce discrimination. We extend this idea by introducing a novel soft resampling procedure that probabilistically samples $\hat{Y}$ based on a fairness-aware adjustment of the leaf predictions. Unlike hard relabeling, our approach, explained hereafter, enables smoother control over the fairness-utility trade-off.

Fair leaf resampling: Using the final tree fitted on X to predict Y ($Tree_Y(X)$), we obtain a set of predicted labels Y_l. The discrimination and accuracy of the tree are defined as, disc $:= P(Y_l = 1|S = 0) - P(Y_l = 1|S = 1)$ and acc $:= P(Y_l = Y)$. For each leaf node ℓ in the tree, which currently predicts $y_l \in \{0,1\}$, we can estimate the potential impact on the total discrimination and accuracy of the tree, if the leaf's label were flipped to $y_l' = 1 - y_l$ as: $\Delta\text{disc}_\ell := P(y_l' = 1|S = 0, X \in \ell) - P(y_l' = 1|S = 1, X \in \ell)$, and $\Delta\text{acc}_\ell := P(y_l' \neq Y, X \in \ell)$.

Thus, the task of producing a fair tree can be formulated as an optimization problem: find a subset of leaf nodes $\mathcal{L}$ whose relabeling, under a discrimination threshold $thr_{disc} = 0$, minimizes the overall discrimination (fairness control - R2) while incurring a minimal loss in accuracy (utility preservation - R3). Formally, the fair resampling objective is:

$$\min_{\mathcal{L}} \left(-\sum_{\ell \in \mathcal{L}} \Delta\text{acc}_\ell \right)$$

with $\text{new\_disc}(\mathcal{L}) := \text{disc} + \sum_{\ell \in \mathcal{L}} \Delta\text{disc}_\ell \leq \text{thr}_{disc}$

As proven in [27], finding $\mathcal{L}$ can be reduced to a *KNAP-SACK* problem, allowing the use of a greedy algorithm (Algorithm 2-i), based on the discrimination-to-accuracy ratio of

---

[2]In future work, we plan to incorporate alternative fairness notions.

Algorithm 2 Fair Leaf Resampling

> **Input** $Tree$, λ (fairness/utility tradeoff), $thr_{disc} = 0$

> **Output** Fair $Tree$ ▷ Tree with fair leaf sampling probabilities p_ℓ

> **i) Greedy leaf search**
> 1: $\mathcal{C} = \{\ell \in Tree \mid \Delta\text{disc}_\ell < 0\}$ ▷ Leaf candidates for relabeling
> 2: $\mathcal{L} = \{\}$
> 3: **while** new_disc($\mathcal{L}$) > thr_{disc} **do**
> 4:   best$_\ell$:= $argmax_{\ell \in \mathcal{C}\backslash\mathcal{L}}(\text{disc}_\ell/\text{acc}_\ell)$
> 5: $\mathcal{L} \leftarrow \mathcal{L} \cup$ best$_\ell$
> 6: **end while**

> **ii) Resampling**
> 7: **for** $\ell \in \mathcal{L}$ **do**
> 8: $p_\ell \leftarrow p_\ell * (1 - \lambda) + (1 - p_\ell) * \lambda$ ▷ Set new sampling probabilities for leaf
> 9: **end for**
> 10: **Return** $Tree$

TABLE I: Dataset characteristics.

Dataset	Num. Samples	Num. Feat. (Num/Cat)	Sensitive Attribute S	Target Class Y	
Adult Census	45k	4/8	sex	income	
				< 50K	≥ 50K
			female	28.80%	3.69%
			male	46.41%	21.09%
Dutch Census	60k	0/11	sex	occupation level	
				low	high
			female	33.71%	16.39%
			male	18.68%	31.21%
Bank Marketing	40k	7/9	marital-st.	deposit	
				no	yes
			married	61.14%	6.88%
			non-married	27.19%	4.77%
KDD Census	95k	7/31	sex	income	
				low	high
			female	50.81%	1.21%
			male	43.42%	4.54%
ACS-I Utah	19k	2/7	sex	income	
				low	high
			female	32.5%	13.74%
			male	23.78%	29.96%
ACS-I Alabama	24k	2/7	sex	income	
				low	high
			female	34.08%	14.31%
			male	26.17%	25.42%

each leaf. However, when generating synthetic data from the leaves, we do not use the leaf's labels y_l; instead, we sample from the output probability distribution p_ℓ (Algorithm 1 line 17). Therefore, for each leaf in $\mathcal{L}$, instead of performing a hard relabeling, we compute adjusted sampling probabilities that reflect a trade-off between fairness and utility (Algorithm 2-ii). This trade-off is governed by a user-defined parameter $\lambda \in [0,1]$, commonly used in fair data generation. For example, consider a selected leaf $\ell \in \mathcal{L}$) with original output probabilities: $p_\ell = \{y_l = 1 : 0.8, y_l = 0 : 0.2\}$. For a complete switch ($\lambda = 1.0$), the new probabilities will be: $p'_\ell = \{y_l = 1 : 0.0, y_l = 0 : 1.0\}$. For a softer adjustment ($\lambda = 0.3$), the new probabilities will be: $p'_\ell = \{y_l = 1 : 0.62, y_l = 0 : 0.38\}$. Adjusted probabilities are then used to generate the target $\hat{Y}$.

To conclude, our fairness-aware target generation method selectively resamples a greedily chosen subset of DT leaves using a soft adjustment mechanism that balances fairness (R2) and utility (R3). This contrasts with approaches such as TabularARGN which apply adjustments across the entire probability distribution and all subgroups, potentially impacting data utility more broadly.

IV. EXPERIMENTS

A. Experimental Setup

To ensure robustness, we perform 3-fold data splits, thus having 66% of the data for training and 33% for testing on unseen real data. These splits are performed at the very beginning of the pipeline, meaning that, for each fold, all generative models are learned on 66% of the whole data and that the real data test set for the current split remains unseen, even during the generation. We found that variability across different splits was much more significant than variability from repeating generative runs within a split, so we use one run per split and report average results with standard deviation across the three folds. Furthermore, splits are kept the same for all generative methods, ensuring that all models have been trained and tested on the same real data.

As stated before, TABFAIRGDT uses the original order of features from the given dataset except when mentioned otherwise. For downstream evaluation, we use LightGBM [32], a SOTA gradient boosting model for tabular data classification, though results remain consistent with other classifiers (not reported due to space constraints). Runtime experiments are conducted on a CPU for our approach, while methods supporting GPU acceleration are run on a GPU. Our hardware specifications include a 12th Gen Intel(R) Core(TM) i9 processor CPU and an Nvidia GeForce RTX 3080 Ti GPU.

B. Datasets

For our experimental evaluation, we selected six publicly available tabular datasets based on relevant surveys [16], [33] that identify datasets suitable for fairness assessment. We list the characteristics of all datasets in Table I, such as the total number of samples, the number of numerical and categorical features, as well as the distribution of the four subgroups defined by the sensitive attribute, and the target. It is important to mention that the Dutch Census dataset consists exclusively of categorical features. Additionally, the KDD Census dataset is the largest in our experiments, containing the highest number of samples. It is primarily composed of categorical features and exhibits a significant class imbalance, which makes it the most skewed dataset considered for the experiments. The last two datasets, namely ACS-I Utah and Alabama, are derived from the *new Adult* dataset [34]. These states were selected by

ranking all states, with an adequate number of samples, based on the difference in positive outcome rates between sensitive groups.

C. Baseline Competitors

We compare TABFAIRGDT with the following state-of-the-art (SOTA) fair generative methods, along with one data augmentation technique. Like our approach, all these methods aim to optimize for downstream statistical parity. We do not compare against tabular data generators that solely optimize for utility, as they do not incorporate fairness objectives.

TabularARGN [14] is a recent, commercially developed, SOTA deep autoregressive method. It first discretizes all features before training the autoregressive model using a random order for the features of each batch. This allows conditional generation, where one or more features may be fixed to given values, and the model generates the remaining feature values. Like our approach, fairness is enforced only at the final step during target generation [7], by aligning the full conditional distributions across sensitive groups. In contrast, our method applies minimal and localized interventions, targeting only a subset of leaves selected greedily by our resampling algorithm.

TabFairGAN [6] is a deep method based on GANs, incorporating fairness in the training process, which is done in two phases. First, the model undergoes normal training to learn the original data distribution, then the loss function is modified to incorporate fairness, optimizing for statistical parity.

CuTS [15] is another deep approach developed to be a customizable generator. Training is performed in two phases: the model is pre-trained to generate data that closely matches the distribution of the real samples, and then a second, user-defined, objective is used to fine-tune the model. Various objectives can be used in the second phase, including fairness, which is relevant to our work, e.g., statistical parity.

FSMOTE [35] is a SMOTE-based [23] data augmentation method that augments minority subgroups from the target and the sensitive attribute to address class and group imbalance.

PreFair [25] uses a deep architecture to generate private and fair data. It leverages differential privacy and causal fairness constraints to prevent unjust causal relationships between the sensitive attribute and the target.

D. Metrics

We evaluate the generated data by measuring (i) utility, i.e. the predictive performance for a downstream task (R3), (ii) fairness, in terms of statistical parity for the downstream task (R2), and (iii) data quality, in terms of distribution and similarity between real and fake samples (R1).

Utility: We train a classifier on the synthetic data $f(\hat{X}) \rightarrow \hat{Y}$, and evaluate on real unseen test data, measuring the ROC AUC score of the predictions Y_{pred}.

Fairness: We use the same trained classifier and real test data to evaluate for fairness, by comparing the conditional probabilities between subgroups of S in the test set. Specifically, we measure statistical parity, defined as:

$$\text{Stat. Par.} := P(Y_{pred} = 1 | S = 0) - P(Y_{pred} = 1 | S = 1)$$

Our goal is to minimize this difference, ensuring that the predicted positive outcome rates are the same across subgroups.

Data quality: First, we train a classifier on $D \cup \hat{D}$ to distinguish between real and synthetic samples. The quality of the generated data is then evaluated using the ROC AUC score, which serves as a detection metric. Ideally, we aim for a detection score (Det. Sc.) close to 0.5, indicating that the classifier struggles to differentiate between real and synthetic samples. Furthermore, we study the overall quality of the generated data by comparing their statistical distribution with the original samples. We use the Kolmogorov-Smirnov (KS) score to evaluate this with continuous features, which computes the maximal difference between the cumulative density functions of the real and synthetic data, and the Total Variation (TV) score for categorical features, which compares the probabilities for all the categorical values of a feature. In addition, we consider precision, recall [36], density, and coverage [37] scores. All four of these metrics use neighborhood spheres computed using the closest data point. Precision (Prec.) gets how many of the synthetic data fall in at least one real sphere, while recall (Rec.) computes the inverse, how many real data fall in at least one synthetic sphere. Density (Dens.) is a variation of precision to take outliers into account. It computes how many of the real spheres a synthetic point appears in. Finally, coverage (Cov.) is a variation of recall and calculates the fraction of the real neighborhood spheres that contain a synthetic data point. As these last four metrics may be fooled by generated samples being duplicates of the real ones, we also consider the median Distance to Closest Record (DCR) [38]. This metric uses nearest neighbors to compute the ratio between median distances from generated samples to real neighboring samples and from the real samples to their own nearest neighbors. DCR ranges from 0 and is unbounded from above, with values smaller than 1 indicating that the generated data are closer to the real ones, while larger values indicate generated data further away from the real samples.

V. RESULTS

We begin by examining the fairness-utility tradeoff (requirements R2-R3 described in Section III-A). Specifically, we compare our method, TABFAIRGDT, to training on the real data, and to two competing deep models, TabularARGN and TabFairGAN, both of which also incorporate a fairness-utility tradeoff parameter $\lambda \in [0, 1]$. Figure 2 showcases how this tradeoff parameter affects both utility (ROC AUC) and fairness (Stat. Par.). All three methods perform mostly as anticipated, balancing utility retention with a reduction in discrimination as λ increases. However, TabFairGAN fails for the KDD Census and Bank Marketing datasets, which are higher-dimensional, and highly class-imbalanced (see Section IV-B). In fact, it results in worse fairness compared to training on real data, as shown in Table IV. Consequently, we exclude its plot for these datasets. We highlight that, even in this highly imbalanced case, our method can improve fairness with a minor loss in utility. Additionally, for the Dutch census dataset, which consists solely of discrete features, TABFAIRGDT is the only
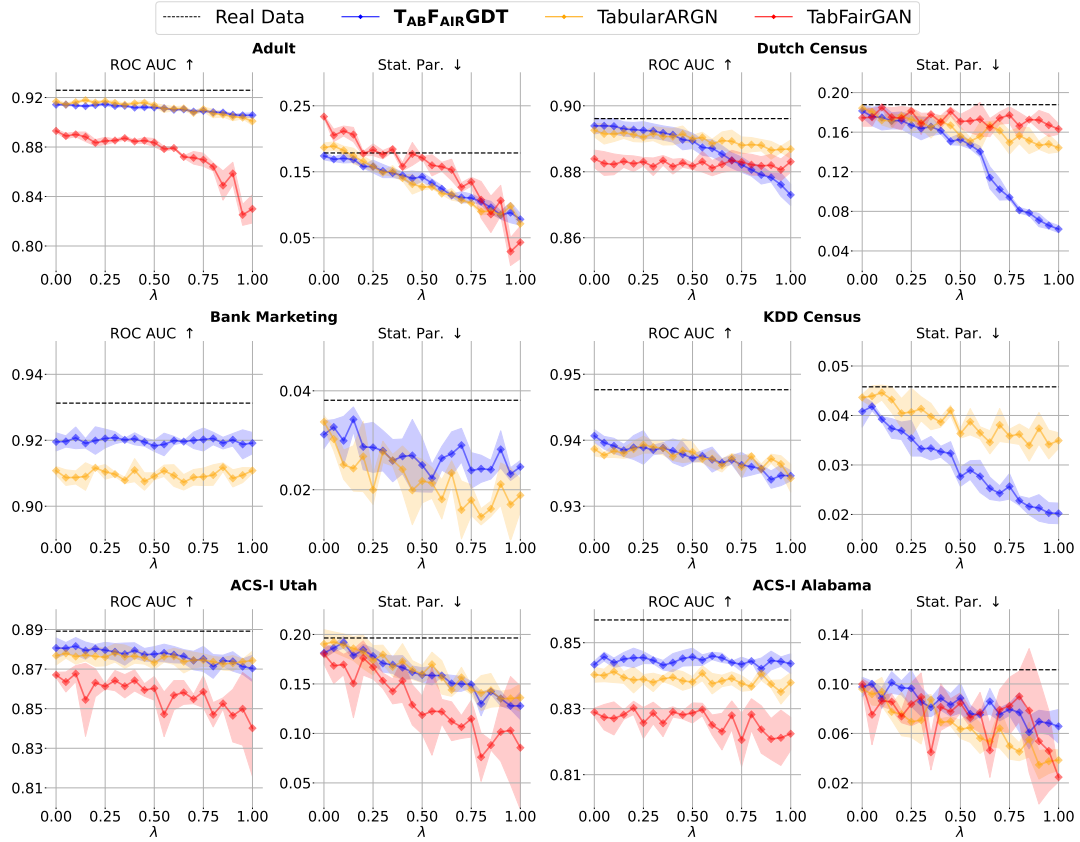
Fig. 2: Impact of the fairness-utility tradeoff parameter λ on model performance across all datasets. As λ increases, models prioritize fairness, often at the expense of utility.

TABLE II: Percentile difference of utility and fairness compared to training on real data, for all methods, averaged across all datasets. Methods that fail in at least one dataset are indicated in red.

Method	Utility ROC AUC ↑	Fairness Stat. Par. ↓
TABFAIRGDT	**-1.85%** $\pm$ 0.47%	**-48.41%** $\pm$ 12.03%
TabularARGN	-1.87% $\pm$ 0.56%	-42.28% $\pm$ 17.17%
TabFairGAN	-9.53% $\pm$ 7.15%	+26.47% $\pm$ 132.57%
CuTS	-5.57% $\pm$ 4.55%	-44.59% $\pm$ 23.69%
FSMOTE	-0.50% $\pm$ 0.12%	-24.58% $\pm$ 5.38%
PreFair	-15.71% $\pm$ 3.19%	-66.55% $\pm$ 14.40%

method that achieves a notable fairness improvement, while loosing only 2% of utility. This further demonstrates the advantage of our method's non-parametric approach, which offers greater flexibility and can handle discrete features with no need for pre-processing. For the remainder of our results, we set the utility-tradeoff hyperparameter to $\lambda = 1$ (for optimal fairness) for all methods.

Table II provides a summary of our results, averaged across all datasets, in terms of percentile utility loss and fairness improvement, compared to training on real data. The more detailed per-dataset results are presented in Table IV. To ensure a fair comparison, the aggregated results of Table II

exclude failed cases, i.e. where a method results in a utility loss greater than 30% or fails to improve statistical parity relative to real data. We highlight in red the methods that fail on at least one dataset. From the results, we observe that our method, TABFAIRGDT, leads to a major improvement in fairness, reducing statistical parity by almost 50% on average, while losing less than 2% in utility. Moreover, TABFAIRGDT is more consistent across datasets, with a standard deviation of only 12% for fairness, compared to much higher variability observed in other methods. TabFairGAN, in particular, shows exceptionally high standard deviation, indicating inconsistent performance across datasets. This suggests its effectiveness is highly sensitive to dataset characteristics, limiting practical reliability. Furthermore, our method, together with TabularARGN, are the only ones that succeed in producing realistic and fair synthetic data for all datasets. Although PreFair performs good for fairness, it also results in large drops in utility. Notably, as observed in Table IV, our method is the only one that succeeds in improving fairness for the Dutch Census dataset, which has only categorical features. We attribute this success to our non-parametric modeling approach.

We continue our evaluation by focusing on generated data quality (requirement R1). As reported in Table III, our method is the clear winner, achieving a detection score of only 0.54%. For the other metrics, TABFAIRGDT demonstrates optimal

TABLE III: Results for data quality metrics averaged across all datasets

Method	Det. Sc. ≈ 0.50	KS ↑	TV ↑	Precision ↑	Recall ↑	Density ↑	Coverage ↑	DCR ≈ 1.00
TABFAIRGDT	**0.54** ±0.0	**0.83** ±0.4	**0.99** ±0.0	0.82 ±0.2	0.82 ±0.2	**0.79** ±0.3	**0.82** ±0.2	**1.02** ±0.0
TabularARGN	0.60 ±0.0	0.82 ±0.4	0.98 ±0.0	0.81 ±0.2	**0.84** ±0.2	0.73 ±0.3	0.81 ±0.2	1.27 ±0.2
TabFairGAN	0.83 ±0.1	0.75 ±0.4	0.96 ±0.0	0.61 ±0.3	0.71 ±0.3	0.48 ±0.3	0.57 ±0.3	2.44 ±1.1
CuTS	1.00 ±0.0	0.49 ±0.3	0.62 ±0.4	0.28 ±0.4	0.31 ±0.5	0.24 ±0.4	0.27 ±0.4	18.21 ±25.0
FSMOTE	0.64 ±0.1	0.77 ±0.4	0.95 ±0.0	**0.83** ±0.2	0.82 ±0.2	0.76 ±0.3	0.79 ±0.2	1.08 ±0.2
PreFair	0.93 ±0.1	0.71 ±0.4	0.97 ±0.0	0.55 ±0.2	0.77 ±0.2	0.34 ±0.2	0.44 ±0.2	4.40 ±2.1

TABLE IV: Results per dataset for all methods. Failures are indicated in red color.

Method	ROC AUC ↑	Stat. Par. ↓	ROC AUC ↑	Stat. Par. ↓
	Adult		Dutch Census	
Real Data	0.926 ± 0.001	0.178 ± 0.007	0.896 ± 0.003	0.188 ± 0.005
TABFAIRGDT	0.906 ± 0.002	0.078 ± 0.009	0.873 ± 0.003	0.062 ± 0.004
TabularARGN	0.901 ± 0.003	0.071 ± 0.007	0.887 ± 0.003	0.144 ± 0.004
TabFairGAN	0.830 ± 0.011	0.043 ± 0.025	0.883 ± 0.003	0.163 ± 0.005
CuTS	0.905 ± 0.003	0.145 ± 0.006	0.500 ± 0.000	-
FSMOTE	0.919 ± 0.001	0.225 ± 0.003	0.893 ± 0.003	0.153 ± 0.008
PreFair	0.760 ± 0.012	0.057 ± 0.012	0.804 ± 0.003	0.105 ± 0.001
	Bank Marketing		KDD Census	
Real Data	0.931 ± 0.001	0.038 ± 0.003	0.948 ± 0.001	0.046 ± 0.002
TABFAIRGDT	0.919 ± 0.003	0.025 ± 0.001	0.935 ± 0.001	0.020 ± 0.002
TabularARGN	0.911 ± 0.001	0.019 ± 0.003	0.934 ± 0.002	0.035 ± 0.002
TabFairGAN	0.716 ± 0.012	0.071 ± 0.017	0.827 ± 0.028	0.181 ± 0.036
CuTS	0.896 ± 0.001	0.016 ± 0.004	0.925 ± 0.002	0.008 ± 0.001
FSMOTE	0.919 ± 0.003	0.051 ± 0.003	0.937 ± 0.001	0.060 ± 0.001
PreFair	0.511 ± 0.017	-	0.613 ± 0.118	-
	ACS-I Utah		ACS-I Alabama	
Real Data	0.889 ± 0.004	0.196 ± 0.008	0.857 ± 0.001	0.111 ± 0.005
TABFAIRGDT	0.870 ± 0.007	0.128 ± 0.014	0.844 ± 0.003	0.066 ± 0.008
TabularARGN	0.874 ± 0.005	0.136 ± 0.014	0.838 ± 0.003	0.038 ± 0.008
TabFairGAN	0.840 ± 0.024	0.086 ± 0.062	0.822 ± 0.005	0.025 ± 0.004
CuTS	0.845 ± 0.008	0.148 ± 0.043	0.733 ± 0.029	0.069 ± 0.052
FSMOTE	0.885 ± 0.005	0.150 ± 0.010	0.851 ± 0.001	0.076 ± 0.010
PreFair	0.729 ± 0.041	0.032 ± 0.029	0.715 ± 0.029	0.033 ± 0.011

TABLE V: Computational runtime for fitting, sampling, and total time (in seconds)

(a) For all methods on a dataset of 10 features and 10k samples

Method	Fitting	Sampling	Total
TABFAIRGDT	0.81	0.25	**1.05**
TabularARGN	10.72	0.12	10.84
TabFairGAN	71.74	**0.008**	71.75
CuTS	882.44	0.20	882.65
FSMOTE	0.018	29.17	29.19
PreFair	**0.005**	45.49	45.5

(b) For TABFAIRGDT vs TabularARGN on varying dataset sizes

Dataset size		TABFAIRGDT			TabularARGN		
Num. Feat.	Num. Samp.	Fit.	Samp.	Tot.	Fit.	Samp.	Tot.
10	1k	**0.15**	**0.06**	**0.21**	10.88	0.09	10.97
	10k	**0.81**	0.25	**1.05**	10.72	**0.12**	10.84
	50k	**3.84**	1.04	**4.88**	27.18	**0.33**	27.51
100	1k	**2.76**	1.03	**3.79**	36.75	**0.75**	37.50
	10k	**14.27**	3.70	**17.97**	36.37	**1.15**	37.52
	50k	**70.11**	15.88	**85.99**	126.71	**3.25**	129.96
500	1k	**33.43**	18.48	**51.91**	206.90	**9.38**	216.28
	10k	**167.08**	43.29	**210.37**	580.68	**13.78**	594.46
	50k	**897.80**	174.12	**1071.92**	2936.21	**31.74**	2967.95
Average speedup (%)		**+77** ±16	-188 ±147	**+72** ±19			

scores and generally surpasses the other deep models. Furthermore, the average DCR score of 1.02 indicates that the synthetic data are as close to real samples as real samples are to each other, showing no signs of overfitting (which would correspond to a DCR below 1). We attribute these strong data quality results to the use of DTs, which effectively capture the structure of the training data, enabling the generation of synthetic data that closely aligns with the original distribution. Additionally, our method requires no pre-processing, unlike TabularARGN, which relies on discretization and rare-feature handling that can impact quality on certain datasets.

The results of our method can be summarized by considering the three requirements of fair synthetic data, listed in Section III-A. First, TABFAIRGDT can generate data that closely match the original samples, as shown by the metrics for data quality in Table III (**R1**). Next, our method shows strong performance in reducing statistical parity for all datasets (Table II and Table IV), progressing towards statistical independence between the target and the sensitive attribute (**R2**). Finally, TABFAIRGDT achieves this performance while having a minor impact on utility scores, as shown in Table II (**R3**). Our method thus meets all three requirements needed to obtain fair synthetic data while using simple DTs. Moreover, it performs on average better and is more efficient than deep approaches that also meet requirements, such as TabularARGN, without the need for specialized hardware.

A. Computational Runtimes

Our method is not only effective but also highly efficient and lightweight, operating entirely on the CPU. We evaluate all approaches on a medium-sized synthetic dataset with 10 features and 10k instances and report fitting, sampling, and total runtime in Table Va. To ensure a fair comparison, we utilize GPU acceleration for methods that support it. Our results show that TABFAIRGDT achieves the fastest total time, generating synthetic data in one second, followed by TabularARGN. Given that TabularARGN is a commercially developed product where efficiency is a priority, it incorporates an early stopping criterion during the fitting phase. Consequently, we select TabularARGN as the primary benchmark for further comparisons across varying dataset sizes, increasing both the number of features and instances, presented in Table Vb. Once again, our method consistently demonstrates the lowest total time, outperforming TabularARGN by an average of 72%. This advantage is primarily due to our highly efficient fitting process, which enables parallel tree construction. Notably, fast fitting can be particularly valuable in real-world scenarios
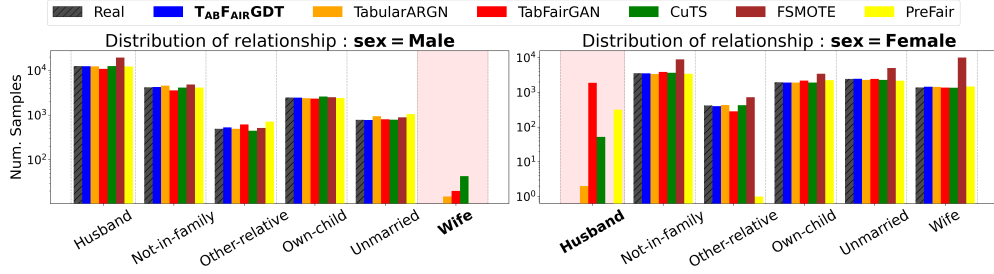
Fig. 3: Per-group feature distributions of real vs. synthetic data for the *relationship* attribute of the Adult census dataset. Areas with synthetic OOD samples are highlighted in red. Our method TᴀʙFᴀɪʀGDT does not produce any OOD samples.
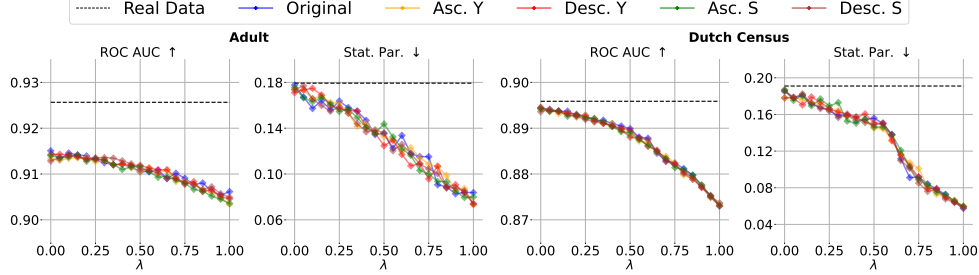

Fig. 4: Effect of column ordering in TᴀʙFᴀɪʀGDT on the utility-fairness tradeoff. Standard deviations were removed for clarity.

such as continuous model retraining during deployment, where low-latency updates are critical. However, as the dataset size increases, our method experiences slower sampling times due to the inherently sequential nature of the generation process.

B. Generation of Out-Of-Distribution Samples

We analyze data from all methods to detect sensitive attribute-specific out-of-distribution (OOD) samples. Using the Adult census dataset as an example, we examine the *relationship* feature by comparing real and synthetic distributions, per-group, in Figure 3. Notably, real data does *not* contain instances where {sex=Male, rel.=Wife} or {sex=Female, rel.=Husband}. However, all baseline generative models introduce such OOD synthetic samples. Specifically, GAN-based methods TabFairGAN, CuTS, and PreFair, which first train a deep generative model and then fine-tune for fairness, generate a significant number of these instances. TabularARGN, despite using autoregressive modeling with fairness constraints applied only at the final step, also produces such samples, likely due to its random feature reordering. In contrast, our approach prevents this by generating $\hat{S}$ only after $\hat{X}$ is fully synthesized, and enforces fairness only for the target generation, not for features X_j. This prevents sampling from a distribution where $X_j \perp S$, thus respecting real-world constraints and avoiding unrealistic feature combinations. Interestingly, FSMOTE also avoids this problem by generating synthetic samples via neighborhood-based interpolation.

Such OOD feature combinations might improve fairness by confusing downstream classifiers, but can be problematic in critical domains like healthcare. We argue that in scenarios involving subgroup-specific medical conditions (e.g., pregnancy, testicular cancer, etc.), it is crucial to ensure fairness

without introducing OOD samples, especially when making the synthetic data publicly available.

C. Impact of Feature Ordering in TᴀʙFᴀɪʀGDT

Due to its autoregressive architecture at both fitting and generating times, one could argue that the order in which the features are considered may impact the results obtained with TᴀʙFᴀɪʀGDT. The experiment in this subsection analyzes this factor by examining five possible orders: 1. the original order of the features in the dataset, 2, 3. ascending/descending order based on the correlation between the features and the target (Asc./Desc. Y), and 4, 5. ascending/descending order based on the correlation between the features and the protected attribute (Asc./Desc. S). Figure 4 shows that all feature orders produce nearly identical results, with no statistically significant differences observed. This trend was consistent across other datasets, and no meaningful changes were found in data quality metrics, so they are omitted here. We conclude that feature order does not affect the results obtained with TᴀʙFᴀɪʀGDT.

VI. CONCLUSION AND FUTURE WORK

We introduced TᴀʙFᴀɪʀGDT, a method for generating fair synthetic data using autoregressive DTs. TᴀʙFᴀɪʀGDT produces realistic data, mitigates discrimination in downstream tasks, and maintains strong predictive performance. Our method combines the simplicity and efficiency of DTs, outperforming deep generative models in terms of data quality, utility, and fairness. Furthermore, TᴀʙFᴀɪʀGDT is up to 72% faster on average, generating fair data for medium-sized datasets in just one second on a standard CPU.

Future work includes extending our framework to support additional fairness definitions, intersectional fairness, regres-

sion tasks, and continuous sensitive attributes, as well as exploring hyperparameters, such as tree depth, to examine how over/under-fitting influences fairness and data quality. We also plan to use dataset similarity measures, e.g., the DT-partition-based approach in [39], to assess synthetic data quality, and the impact of fairness thresholds on the generated distributions.

Acknowledgment

References

[1] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A Survey on Bias and Fairness in Machine Learning," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–35, Jul. 2022.

[2] R. Schwartz, A. Vassilev, K. Greene, L. Perine, A. Burt *et al.*, *Towards a standard for identifying and managing bias in artificial intelligence*. US Department of Commerce, National Institute of Standards and Technology, 2022, vol. 3.

[3] L. Bothmann, K. Peters, and B. Bischl, "What Is Fairness? On the Role of Protected Attributes and Fictitious Worlds," 2024, arXiv:2205.09622.

[4] S. Caton and C. Haas, "Fairness in Machine Learning: A Survey," *ACM Computing Surveys*, 3616865, p. 3616865, 2024.

[5] M. Chui, J. Manyika, M. Miremadi *et al.*, "Notes from the AI frontier: Insights from hundreds of use cases," *McKinsey Global Institute*, vol. 2, no. 267, pp. 1–31, 2018.

[6] A. Rajabi and O. O. Garibay, "Tabfairgan: Fair tabular data generation with generative adversarial networks," *Machine Learning and Knowledge Extraction*, vol. 4, no. 2, pp. 488–501, 2022.

[7] I. Krchova, M. Platzer, and P. Tiwald, "Strong statistical parity through fair synthetic data," *arXiv preprint arXiv:2311.03000*, 2023.

[8] C. Leininger, S. Rittel, and L. Bothmann, "Overcoming fairness trade-offs via pre-processing: A causal perspective," *arXiv preprint arXiv:2501.14710*, 2025.

[9] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 259–268.

[10] A. Tawakuli and T. Engel, "Make your data fair: A survey of data preprocessing techniques that address biases in data towards fair AI," *Journal of Engineering Research*, 2024.

[11] R. Sonoda, "Fair oversampling technique using heterogeneous clusters," *Information Sciences*, vol. 640, p. 119059, 2023.

[12] B. Ronval, S. Nijssen, and L. Bothmann, "Can generative AI-based data balancing mitigate unfairness issues in machine learning?" in *Third European Workshop on Algorithmic Fairness, 2024*, 2024.

[13] E. Panagiotou, A. Roy, and E. Ntoutsi, "Synthetic tabular data generation for class imbalance and fairness: A comparative study," *arXiv preprint arXiv:2409.05215*, 2024.

[14] P. Tiwald, I. Krchova, A. Sidorenko, M. Vargas-Vieyra, M. Scriminaci *et al.*, "TabularARGN: A flexible and efficient auto-regressive framework for generating high-fidelity synthetic data," *arXiv preprint arXiv:2501.12012*, 2025.

[15] M. Vero, M. Balunović, and M. Vechev, "Cuts: Customizable tabular synthetic data generation," *arXiv preprint arXiv:2307.03577*, 2023.

[16] T. Le Quy, A. Roy, V. Iosifidis, W. Zhang, and E. Ntoutsi, "A survey on datasets for fairness-aware machine learning," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 12, no. 3, p. e1452, 2022.

[17] S. Verma and J. Rubin, "Fairness Definitions Explained," in *Proceedings of the International Workshop on Software Fairness*. Gothenburg Sweden: ACM, 2018.

[18] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[19] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "Tabddpm: Modelling tabular data with diffusion models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 564–17 579.

[20] H. Zhang, J. Zhang, B. Srinivasan, Z. Shen, X. Qin *et al.*, "Mixed-type tabular data synthesis with score-based diffusion in latent space," *arXiv preprint arXiv:2310.09656*, 2023.

[21] R. McKenna, B. Mullins, D. Sheldon, and G. Miklau, "Aim: An adaptive and iterative mechanism for differentially private synthetic data," *arXiv preprint arXiv:2201.12677*, 2022.

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

[23] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[24] J. P. Reiter, "Using CART to generate partially synthetic public use microdata," *Journal of Official Statistics*, vol. 21, no. 3, p. 441, 2005.

[25] D. Pujol, A. Gilad, and A. Machanavajjhala, "Prefair: Privately generating justifiably fair synthetic data," *arXiv preprint arXiv:2212.10310*, 2022.

[26] S. Yan, H.-t. Kao, and E. Ferrara, "Fair class balancing: Enhancing model fairness without observing sensitive attributes," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1715–1724.

[27] F. Kamiran, T. Calders, and M. Pechenizkiy, "Discrimination aware decision tree learning," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 869–874.

[28] W. Zhang and E. Ntoutsi, "FAHT: an adaptive fairness-aware decision tree classifier," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 1480–1486.

[29] N. Jovanović, M. Balunovic, D. I. Dimitrov, and M. Vechev, "Fare: Provably fair representation learning with practical certificates," in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 401–15 420.

[30] A. K. Menon and R. C. Williamson, "The cost of fairness in binary classification," in *Conference on Fairness, Accountability and Transparency*. PMLR, 2018, pp. 107–118.

[31] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural autoregressive distribution estimation," *Journal of Machine Learning Research*, vol. 17, no. 205, pp. 1–37, 2016.

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[33] X. Han, J. Chi, Y. Chen, Q. Wang, H. Zhao *et al.*, "FFB: A fair fairness benchmark for in-processing group fairness methods," *arXiv preprint arXiv:2306.09468*, 2023.

[34] F. Ding, M. Hardt, J. Miller, and L. Schmidt, "Retiring Adult: New datasets for fair machine learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[35] J. Chakraborty, S. Majumder, and T. Menzies, "Bias in machine learning software: Why? how? what to do?" in *Proceedings of the 29th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*, 2021, pp. 429–440.

[36] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly, "Assessing generative models via precision and recall," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[37] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7176–7185.

[38] A. Danholt Lautrup, T. Hyrup, A. Zimek, and P. Schneider-Kamp, "SynthEval: A framework for detailed utility and privacy evaluation of tabular synthetic data," *arXiv e-prints*, pp. arXiv–2404, 2024.

[39] I. Ntoutsi, A. Kalousis, and Y. Theodoridis, "A general framework for estimating similarity of datasets and decision trees: exploring semantic similarity of decision trees," in *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 2008, pp. 810–821.