Unmasking Trees for Tabular Data

Calvin McCarter
mccarter.calvin@gmail.com
BigHat Biosciences

Abstract

Despite much work on advanced deep learning and generative modeling techniques for tabular data generation and imputation, traditional methods have continued to win on imputation benchmarks. We herein present UnmaskingTrees, a simple method for tabular imputation (and generation) employing gradient-boosted decision trees which are used to incrementally unmask individual features. This approach offers state-of-the-art performance on imputation, and on generation given training data with missingness; and it has competitive performance on vanilla generation. To solve the conditional generation subproblem, we propose a tabular probabilistic prediction method, BaltoBot, which fits a *bal*anced *t*ree *o*f *bo*osted *t*ree classifiers. Unlike older methods, it requires no parametric assumption on the conditional distribution, accommodating features with multimodal distributions; unlike newer diffusion methods, it offers fast sampling, closed-form density estimation, and flexible handling of discrete variables. We finally consider our two approaches as meta-algorithms, demonstrating in-context learning-based generative modeling with TabPFN.[1]

1 Introduction

Given a tabular dataset, it is frequently desirable to impute missing values within that dataset, and to generate new synthetic examples. On data generation, recent work [Jolicoeur-Martineau et al., 2024b] (ForestDiffusion) has shown state-of-the-art results on data generation using gradient-boosted trees [Chen and Guestrin, 2016] trained on diffusion or flow-matching objectives, outperforming deep learning-based approaches. However, this approach tended to struggle on tabular imputation tasks, outperformed by MissForest [Stekhoven and Bühlmann, 2012], an older multiple imputation approach based on random forests [Breiman, 2001].

We address this shortfall by training gradient-boosted trees to autoregressively unmask features in random order, taking as inspiration the benefits of this training objective applied to tabular Transformer models [Gulati and Roysdon, 2024] (TabMT). This autoregressive approach, which we dub UnmaskingTrees, naturally performs conditional generation (i.e. imputation): at inference time, we simply fill in and condition on observed values, autoregressively generating the remaining missing values. This contrasts with tabular diffusion modeling, for which the RePaint inpainting algorithm [Lugmayr et al., 2022] is employed to mediocre effect [Jolicoeur-Martineau et al., 2024b]. Because the predictor for a given feature must condition on varying subsets of the other features, the ability of gradient-boosted trees to handle missing features makes them a natural choice for autoregressive modeling. Hence, we maintain the tree-based approach of Jolicoeur-Martineau et al. [2024b], while replacing their tree-based regressors with our novel tree-based probabilistic predictors, which we turn to next.

While mean-estimating regression models are satisfactory for diffusion, for autoregression we must inject noise, and hence must estimate the entire conditional distribution of each feature. We therefore

[1] Please see `https://arxiv.org/abs/2407.05593` for the latest methods and results.

Table Representation Learning Workshop at NeurIPS 2024.

revisit the long-studied problem of (tabular) probabilistic prediction [Le et al., 2005, Meinshausen and Ridgeway, 2006]. Because the conditional distribution is possibly multi-modal, parametric approaches such as XGBoostLSS [März, 2019], NGBoost [Duan et al., 2020], and PGBM [Sprangers et al., 2021] are poor choices for our setting. Meanwhile, quantization of a continuous variable can model its multi-modality, but at the cost of destroying either low-resolution or high-resolution information. A diffusion-based method, Treeffuser Beltran-Velez et al. [2024], was recently proposed to address these problems. However, as a diffusion method, it suffers from slow sampling and is unable to provide closed-form density estimates; furthermore, Treeffuser does not naturally model discrete outcomes. To address these problems, we propose BaltoBot, a *bal*anced *t*ree *o*f *bo*osted *t*rees. For each individual variable, we recursively divide its output space with the kernel density integral (KDI) quantizer [McCarter, 2023] into a "meta-tree" of binary classifiers, which for us are gradient-boosted trees. This allows us to efficiently generate samples and estimate densities, because each sample follows only one path from root to leaf of the meta-tree. Performing regression with hierchical classification proved successful in computer vision object bounding box prediction [Li et al., 2020], but has been surprisingly underexplored in tabular ML.

Our two methods are in fact meta-algorithms that, in combination, can create a generative model out of *any* probabilistic binary classifier. To demonstrate this flexibility, we swap out XGBoost [Chen and Guestrin, 2016] for TabPFN [Hollmann et al., 2022]. TabPFN is a deep learning model pretrained to perform in-context learning for tabular classification. While it has state-of-the-art classification benchmark performance [McElfresh et al., 2024], it currently does not perform regression tasks, nor does it inherently perform generative modeling [Ma et al., 2024]. Constructing a generative model out of TabPFN [Hollmann et al., 2022] was first proposed in TabPFGen [Ma et al., 2024], which approximates the posterior from TabPFN-provided likelihoods by iteratively applying stochastic gradient Langevin dynamics [Welling and Teh, 2011]. But unlike the previous work, ours requires only a few TabPFN forward-passes for each sample rather than many iterative data updates.

We showcase UnmaskingTrees on two tabular case studies, and on the benchmark of 27 tabular datasets presented by Jolicoeur-Martineau et al. [2024b]. Most notably on this benchmark, our approach offers state-of-the-art performance on imputation and on generation given training data with missingness; and it has competitive performance on vanilla generation. We also demonstrate that BaltoBot is on its own a useful method for probabilistic prediction, showing its advantages on synthetic case studies. Finally, we provide open-source code with an easy-to-use sklearn-style interface at `https://github.com/calvinmccarter/unmasking-trees`. In addition to being a useful method for practitioners, we hope our work sparks conversations within the tabular ML community about whether diffusion is really all you need for tabular conditional generation.

2 Method

2.1 UnmaskingTrees for tabular joint distribution modeling

UnmaskingTrees combines the gradient-boosted trees of ForestDiffusion [Jolicoeur-Martineau et al., 2024b] with the training objective of TabMT [Gulati and Roysdon, 2024], inheriting the benefits of both. Consider a dataset with N examples and D features. For each example, we generate new training samples by randomly sampling an order over the features, then incrementally masking the features in that random order. Given duplication factor K, we repeat this process times with K different random permutations. This leads to a training dataset with KND samples, given which we train XGBoost [Chen and Guestrin, 2016] models to predict each unmasked sample given the more-masked example derived from it. Implementing this is very simple: it requires about 70 lines of excessively-loquacious Python code for training, and about 20 lines for inference.

For both generation and imputation, we generate features of each sample in random order. For imputation rather than generation tasks, we begin by filling in each sample with the observed values, and run inference on the remaining unobserved features.

2.2 BaltoBot for tabular probabilistic prediction

A key problem when autoregressively generating continuous data is that a regression model will attempt to predict the mean of a conditional distribution, whereas we would like it to sample from the possibly-multimodal conditional distribution. The simplest solution is to quantize continuous

features into bins, because classification over histograms is inherently multimodal; TabMT [Gulati and Roysdon, 2024] did this with 1d k-Means clustering [Lloyd, 1982]. Yet this not only destroys information within bins due to rounding, it also destroys information about the proximity among the ordered bins. Thus, it forces us to choose between a small number of quantization bins, yielding low resolution; or to choose a large number of bins, risking catastrophic errors due to overfitting and/or clumping of generated samples due to poor calibration. This not only limits performance, but also necessitates hyperparameter tuning [Gulati and Roysdon, 2024].

Inspired by this, we propose a general-purpose solution to the tabular probabilistic prediction problem. For each individual regression output variable, we build a balanced tree of binary classifiers. Consider a node with depth δ on this "meta-tree", which is fit with $(\mathbf{X}_{\text{train}} \in \mathbb{R}^{n \times d}, \mathbf{y}_{\text{train}} \in \mathbb{R}^n)$. Using kernel density integral quantization (KDI) [McCarter, 2023], which adaptively interpolates between uniform quantization and quantile quantization, we obtain binarized $\tilde{y}_{\text{train}} \in [0,1]^n$. We train an XGBoost classifier on $(\mathbf{X}_{\text{train}}, \tilde{\mathbf{y}}_{\text{train}})$. If $\delta > 0$, we then recursively pass $\{(\mathbf{x}^{(i)}, y^{(i)}) \in (\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}) | \tilde{y}^{(i)} = 0\}$ to its left child, and analogously for $\tilde{y}^{(i)} = 1$ to its right child. At a leaf node, $\delta = 0$, if given a single unique training set output value in a bin, we record this value. At inference time, given a query input $\mathbf{x}$, we descend the tree by obtaining predicted probabilities from each node's XGBoost classifier, then sampling from these. Once we reach a leaf node, we either sample uniformly from its appropriate bin, or we return the lone output value if a singleton bin.

At training and inference time, each XGBoost model within the meta-tree only sees examples that fall into its corresponding region of the output space. Thus, for a meta-tree with depth Δ (and thus 2^Δ models), each example is only passed as input to Δ different models. While lower-level classifiers receive less data and are poorer quality, the magnitude of such errors are smaller due to our hierarchical partitioning approach. Furthermore, our singleton-bin technique allows us to adaptively generate discrete and even mixed-type variables, if these discrete outcomes are high-frequency relative to the total size of the data and to the depth of the meta-tree. (Up to 2^Δ discrete outcomes can be produced by BaltoBot.) Finally, eschewing diffusion modeling enables us to perform closed-form conditional density estimation.

2.3 Computational complexity

ForestDiffusion, with T diffusion steps and duplication factor K, constructs a training dataset of size $TKN \times D$. Given the same duplication factor K, UnmaskingTrees will construct a training dataset of size $KND \times D$. Meanwhile, ForestDiffusion must train DT different XGBoost regression models. We, on the other hand, train D different BaltoBot models, one per feature; with BaltoBot meta-tree depth of Δ, we then train a total of $D2^\Delta$ XGBoost binary classifiers. However, classifiers lower in the BaltoBot meta-tree become progressively faster to train. Indeed, each constructed training sample will be seen by DT different XGBoost regressors with ForestDiffusion, but only $D\Delta$ classifiers with our approach. Given that $T \sim 50$ and $\Delta \sim 4$, this yields a large speedup for our approach.

The KDI quantizer [McCarter, 2023] has negligible contribution to runtime, because it uses the polynomial-exponential kernel density estimator (KDE) [Hofmeyr, 2019], which has linear complexity in sample size for 1d data, unlike the quadratic complexity of the Gaussian KDE.

At inference time, each ForestDiffusion generated sample passes through T steps of the diffusion reverse-process, for a total of DT XGBoost predictions. For UnmaskingTrees with BaltoBot, each generated sample instead requires only $D\Delta$ XGBoost predictions, because each sample follows only one path from root to leaf of the meta-tree. The resulting speedup is especially impactful for the multiple imputation scenario, where inference time dominates.

2.4 In-context learning-based generation with BaltoBoTabPFN and UnmaskingTabPFN

Within our flexible frameworks for joint and conditional modeling, TabPFN [Hollmann et al., 2022] can be used as a base learner for probabilistic prediction and generative modeling. For UnmaskingTabPFN joint modeling, a difficulty arises from TabPFN's inability to handle inputs $\mathbf{X}_{\text{train}}$ with missing values. To address this, during both training and inference, we replaced NaNs with samples from $\mathcal{N}(0,1)$; we found this performed better than removing samples and/or features containing NaNs. To address TabPFN's sample size limit of 1024, we performed random subsampling without replacement.

3 Results

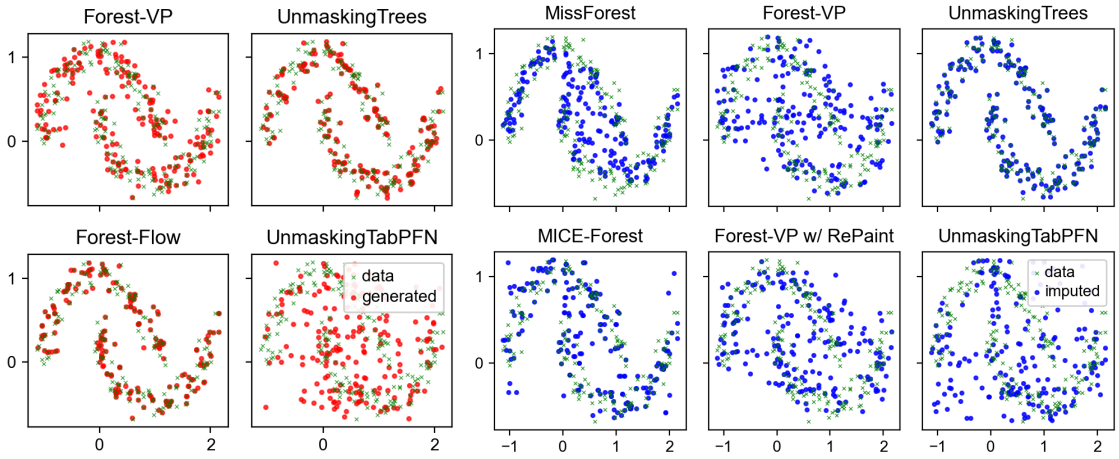

Figure 1: Results on Two Moons case study. Original data is shown in green; generated data is shown in red; imputed data is shown in blue.

We evaluate UnmaskingTrees on two case studies (Section 3.1) and on a tabular benchmark of 27 datasets (Section 3.2); we lastly evaluate BaltoBot on tabular probabilistic prediction (Section 3.3). Results were obtained always using the default hyperparameters: output tree depth of 4, and duplication factor $K = 50$. Our hyperparameters were tuned on the two case study datasets, then applied without further tuning to the benchmark experiment. Overall, UnmaskingTrees has state-of-the-art performance on imputation and on generation after training on incomplete data; and it has competitive performance on vanilla tabular generation scenarios.

3.1 Case studies on Two Moons and Iris datasets

Two Moons dataset We first compare our approach to previous leading methods on the synthetic Two Moons dataset with 200 training samples and noise level $\mathcal{N}(0, 0.1)$. We compare UnmaskingTrees to MissForest [Stekhoven and Bühlmann, 2012], MICE-Forest [Van Buuren et al., 1999, Wilson et al., 2022] (another leading traditional multiple imputation method), and ForestDiffusion, with default hyperparameters for all methods. For ForestDiffusion, we evaluate both the variance-preserving SDE (Forest-VP) and flow-matching (Forest-Flow) versions on generation; on imputation, we evaluate Forest-VP with and without RePaint, again using default RePaint hyperparameters.

We show results in Figure 1. On generation, Forest-VP appears to do best according to visual inspection, while UnmaskingTrees and Forest-Flow perform similarly decently. UnmaskingTabPFN performs poorly, but does capture the overall shape of the distribution. Next, we turn to imputation, wherein we request a single imputation for a copy of the original training data with the second dimension (y-axis) values masked out. ForestDiffusion struggles with and without RePaint, and MissForest and MICE-Forest have a lesser degree of out-of-distribution imputations. Meanwhile, UnmaskingTrees generates impeccable imputations.

Iris dataset In Figure 2, we show results for the Iris dataset [Fisher, 1936], plotting petal length, petal width, and species. We compare both methods on generation, and to compare on imputation, we create another version of the Iris dataset, with missingness completely at random: we randomly select samples with 50% chance to have any missingness, and on these samples, we mask the non-species feature values with 50% chance. Visually, ForestDiffusion and UnmaskingTrees perform about equally well on generation. Meanwhile, on imputation, UnmaskingTrees does a better job conditioning on species information than ForestDiffusion. UnmaskingTrees also produces more diverse imputations than MissForest.

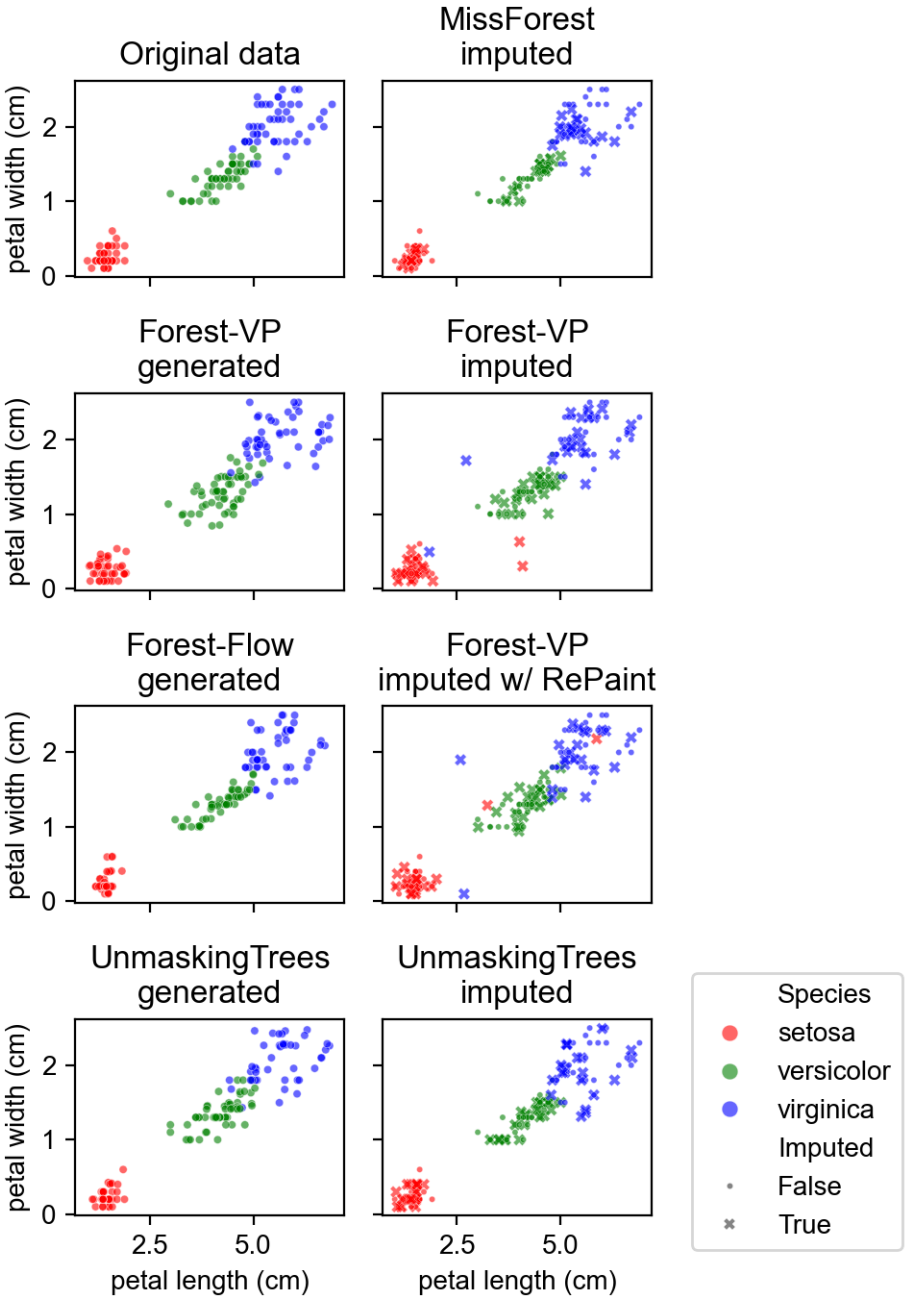

Figure 2: Results on Iris dataset, with species, petal width, and petal length depicted. Original data and syntetically-generated datasets are shown on the left column. The imputed dataset is shown on the right column, with $\times$ symbols highlighting the samples with any missingness that required imputation.

3.2 Benchmarking UnmaskingTrees on 27 tabular datasets

We next repeat the experimental setup of Jolicoeur-Martineau et al. [2024b] for evaluating tabular imputation and generation methods.[2] Results for imputation are shown in Table 1. UnmaskingTrees

[2]We do not compare against TabMT [Gulati and Roysdon, 2024] and TabPFGen [Ma et al., 2024] because no code was provided.

wins first place on 3/9 metrics, including both metrics based on downstream prediction tasks; and it generally outperforms ForestDiffusion, winning on 8/9 metrics. While MissForest wins first place on 4/9 metrics, UnmaskingTrees wins 5-4 head-to-head against MissForest. UnmaskingTrees is also the only method with better than 5th place average ranking on all metrics. We report further ablation experiments in Appendix A, showing progressive improvements for the UnmaskingTrees framework, for KDI quantization versus k-Means, and for the BaltoBot method used in our full proposed solution.

We next repeat the experimental setup of Jolicoeur-Martineau et al. [2024b] for evaluating tabular generation methods. Results for partially-missing data are shown in Table 2. UnmaskingTrees is first place on 5/9 metrics; head-to-head, UnmaskingTrees beats TabDDPM 5-4, and beats Forest-Flow 6-3. Results for fully-observed data are shown in Table 3. UnmaskingTrees loses head-to-head to Forest-Flow, Forest-VP, and TabDDPM, but wins against the other methods.

Table 1: Tabular data imputation (27 datasets, 3 experiments per dataset, 10 imputations per experiment) with 20% missing. Shown are *averaged rank* over all datasets and experiments (standard-error). Overall best is highlighted; better of Forest-VP versus ours is **boldface blue**. Metrics are Minimum and Average mean-absolute error (MinMAE and AvgMAE) to ground-truth, Wasserstein distance to train and test dataset distributions (W_{train} and W_{test}), Mean Absolute Deviation (MAD) around the median/mode (for diversity), R^2 and F_1 for downstream regression / classification problems, and percent bias P_{bias} and confidence interval coverage rate Cov_{rate} for statistical inferences.

	MinMAE ↓	AvgMAE ↓	W_{train} ↓	W_{test} ↓	MAD ↓	R^2 ↓	F_1 ↓	P_{bias} ↓	Cov_{rate} ↓
KNN	5.5 (0.5)	6.3 (0.4)	4.9 (0.4)	5 (0.4)	8.4 (0)	6.5 (1)	5.7 (1.1)	6.2 (1)	5.4 (0.6)
ICE	6.8 (0.4)	4.7 (0.4)	7 (0.5)	7.2 (0.4)	1.6 (0.2)	6.2 (1)	7 (0.6)	5.7 (0.9)	5.3 (0.6)
MICE-Forest	3.9 (0.4)	2.5 (0.4)	2.9 (0.2)	3 (0.2)	3.6 (0.2)	3.7 (1.4)	3.2 (1)	5.5 (1.2)	4.3 (0.6)
MissForest	2.7 (0.5)	4 (0.4)	1.8 (0.3)	2 (0.3)	5.5 (0.2)	3.8 (1.4)	2.5 (0.5)	5.5 (1.5)	3.3 (0.5)
Softimpute	6.7 (0.4)	7.6 (0.4)	7.1 (0.5)	7.3 (0.5)	8.4 (0)	6 (0.9)	7.8 (0.4)	6.3 (0.9)	6.7 (0.4)
OT	5.9 (0.4)	6.1 (0.3)	6 (0.5)	6 (0.5)	3.7 (0.3)	6.2 (0.5)	6.8 (0.6)	5.5 (0.8)	4.8 (0.5)
GAIN	4.7 (0.4)	6.5 (0.3)	6 (0.3)	6 (0.2)	6.9 (0.1)	5.7 (0.8)	5.4 (0.8)	4.7 (1)	5 (0.6)
Forest-VP	5.3 (0.4)	4 (0.5)	5.8 (0.3)	5.1 (0.4)	**3.2** (0.4)	4.5 (0.9)	4.6 (0.8)	3.3 (0.6)	5.5 (0.7)
UTrees	**3.5** (0.5)	**3.2** (0.5)	**3.5** (0.4)	**3.5** (0.5)	3.8 (0.2)	**2.5** (0.6)	**2.2** (0.6)	**2.3** (0.9)	**4.7** (0.6)

Table 2: Tabular data generation with incomplete data (27 datasets, 3 experiments per dataset, 20% missing values), MissForest is used to impute missing data except in Forest-VP, Forest-Flow, and UnmaskingTrees; *averaged rank* over all datasets and experiments (standard-error). Overall best is highlighted; better of Forest-VP versus Forest-Flow versus ours is **boldface blue**.

	W_{train} ↓	W_{test} ↓	cov_{train} ↓	cov_{test} ↓	R^2_{fake} ↓	$F1_{fake}$ ↓	$F1_{disc}$ ↓	P_{bias} ↓	cov_{rate} ↓
GaussianCopula	7 (0.3)	7.1 (0.2)	7.2 (0.3)	7.1 (0.3)	6.3 (0.4)	6.6 (0.3)	6.7 (0.4)	5.5 (1)	7.7 (0.6)
TVAE	5.2 (0.3)	4.9 (0.3)	5.7 (0.3)	5.8 (0.2)	6 (1)	5.8 (0.5)	5.8 (0.4)	8 (0.4)	6.2 (1)
CTGAN	8.3 (0.2)	8.4 (0.2)	8.4 (0.2)	8.3 (0.2)	8.3 (0.3)	8.4 (0.2)	6.5 (0.2)	4.8 (1.2)	7.1 (0.7)
CTABGAN	6.7 (0.4)	6.5 (0.4)	7.1 (0.3)	6.8 (0.3)	7.3 (0.6)	7.1 (0.4)	6.6 (0.3)	7.5 (1)	6.1 (0.6)
Stasy	5.9 (0.2)	6.1 (0.3)	5.3 (0.2)	5.1 (0.3)	5.8 (0.9)	4.4 (0.4)	5.3 (0.4)	3.7 (0.4)	4.6 (1.1)
TabDDPM	3 (0.7)	3.4 (0.7)	2.3 (0.5)	2.9 (0.6)	1.7 (0.3)	3.3 (0.6)	3.9 (0.6)	3.8 (1.2)	2 (0.5)
Forest-VP	3.7 (0.2)	3.2 (0.3)	3.9 (0.2)	3.8 (0.3)	3.2 (0.3)	**2.3** (0.3)	4.2 (0.4)	4.2 (0.8)	4.5 (1.1)
Forest-Flow	3 (0.3)	**2.6** (0.3)	2.6 (0.3)	2.7 (0.2)	**3** (0.7)	3.7 (0.3)	5 (0.5)	3.8 (0.9)	**3.2** (0.8)
UTrees	**2.1** (0.2)	2.8 (0.3)	**2.5** (0.2)	**2.5** (0.2)	3.3 (0.8)	3.5 (0.5)	**1** (0)	**3.7** (0.9)	3.7 (1)

Table 3: Tabular data generation with complete data (27 datasets, 3 experiments per dataset); *averaged rank* over all datasets and experiments (standard-error). Overall best is highlighted; better of Forest-VP versus Forest-Flow versus ours is **boldface blue**.

	W_{train} ↓	W_{test} ↓	cov_{train} ↓	cov_{test} ↓	R^2_{fake} ↓	$F1_{fake}$ ↓	$F1_{disc}$ ↓	P_{bias} ↓	Cov_{rate} ↓
GaussianCopula	7.1 (0.3)	7.2 (0.3)	7.3 (0.3)	7.4 (0.3)	6.2 (0.2)	6.4 (0.3)	7 (0.4)	6.5 (1.1)	7.5 (0.7)
TVAE	5.3 (0.2)	5.1 (0.2)	5.7 (0.2)	5.7 (0.2)	6.5 (0.7)	6 (0.5)	5.5 (0.3)	7.3 (0.6)	6.7 (0.6)
CTGAN	8.4 (0.1)	8.4 (0.2)	8.3 (0.2)	8.1 (0.2)	8.5 (0.2)	8.3 (0.2)	6.7 (0.3)	5.3 (1.1)	7.2 (0.5)
CTAB-GAN+	6.8 (0.3)	6.7 (0.3)	7.2 (0.3)	7.1 (0.3)	6.8 (0.4)	6.9 (0.4)	6.9 (0.3)	7.7 (0.8)	6.7 (0.8)
STaSy	6.1 (0.2)	6.3 (0.2)	5.3 (0.2)	5.4 (0.2)	6 (1.2)	5.1 (0.3)	6.1 (0.3)	4.5 (0.8)	4.2 (1.1)
TabDDPM	3 (0.7)	3.9 (0.6)	2.8 (0.5)	3.4 (0.5)	1.2 (0.2)	3.8 (0.6)	3.2 (0.4)	3 (0.9)	1.4 (0.2)
Forest-VP	3.2 (0.2)	2.8 (0.2)	3.6 (0.3)	3.3 (0.3)	2.8 (0.3)	**2.2** (0.3)	4.3 (0.4)	3.2 (0.9)	3.5 (0.8)
Forest-Flow	**1.9** (0.2)	**1.5** (0.2)	**1.7** (0.2)	**1.8** (0.2)	**2.3** (0.4)	2.4 (0.3)	4.3 (0.4)	**2.8** (0.5)	**2.7** (0.4)
UTrees	3.1 (0.1)	3.1 (0.2)	3.1 (0.2)	2.8 (0.2)	4.7 (0.3)	3.9 (0.3)	**1** (0)	4.7 (0.7)	5.2 (0.9)

3.3 Evaluating BaltoBot on synthetic probabilistic prediction case studies

Wave dataset We compare our approach with Treeffuser [Beltran-Velez et al., 2024] on the "wave" synthetic dataset from Treeffuser [Beltran-Velez et al., 2024], which as shown in Figure 3 is nonlinear,

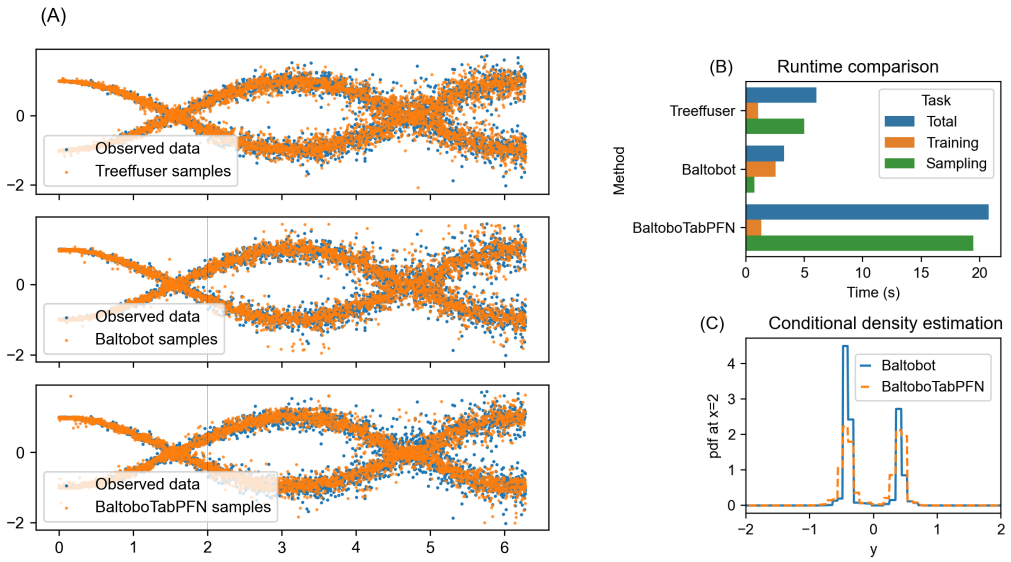

Figure 3: Comparison of Treeffuser and our approach on wave synthetic data with 5000 samples. (A) Probabilistic predictions for Treeffuser (top), BaltoBot (center), and BaltoBoTabPFN (bottom). (B) Runtime comparison for the different methods. (C) Estimated pdf from our methods at $X = 2$, depicted as the vertical dotted line in (A).

multimodal, heteroskedastic, and heavy-tailed. On the raw probabilistic predictions in Figure 3(A), we see that BaltoBot is (by visual inspection) able to model the conditional distribution as well as Treeffuser; BaltoBoTabPFN performs slightly worse. Yet this case study illustrates the two advantages of BaltoBot. First, in Figure 3(B) we show the runtime of the different methods: training, sampling, and total. To train on 5000 samples, Treeffuser took 1.1s and BaltoBot took 2.6s; but to generate 5000 samples, Treeffuser took 5.0s while BaltoBot took 0.72s. Second, BaltoBot offers the ability to estimate a closed-form probability density function (pdf) of the predictive distribution in Figure 3(C); in contrast, Treeffuser can only sample from the predictive distribution.

Poisson-distributed count data We generate 500 samples of $X_i \sim \text{Unif}[0, 3], Y_i \sim \text{Poisson}(\lambda = \sqrt{X_i})$, and show probabilistic predictions for Y in Figure 4. Whereas Treeffuser generates a spurious negative-valued outlier and many non-integer Y samples, our approach automatically models the count-type distribution of the data.

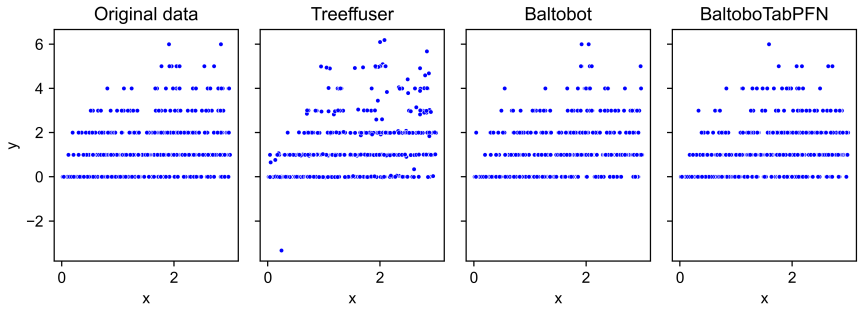

Figure 4: Comparison of Treeffuser, BaltoBot, and BaltoBoTabPFN on Poisson-distributed data. The input variable is on the x-axis, while probabilistic predictions are shown on the y-axis.

4 Discussion

Diffusion modeling has recently gained popularity in tabular ML [Zheng and Charoenphakdee, 2022, Jolicoeur-Martineau et al., 2024b, Beltran-Velez et al., 2024, Kotelnikov et al., 2023]. Our proposed approach is an instance of the autoregressive discrete diffusion framework [Hoogeboom et al., 2021], instances of which have shown success in a variety of tasks [Yang, 2019, Austin et al., 2021, Kitouni et al., 2024, Jolicoeur-Martineau et al., 2024a]. Yet our results call into question whether diffusion is beneficial for tabular conditional generation, or whether autoregression is sufficient for our setting. It has been observed that diffusion is autoregression in frequency space, progressing from low frequencies to high frequencies, which makes it a good match for image data with its power law spectra [Rissanen et al., 2022, Dieleman, 2024, Stews, 2024]. In tabular datasets without this phenomena, we would expect diffusion modeling to be less advantageous. Our success also makes sense in light of the observation that unmasking tends to outperform denoising in self-supervised pretraining [Balestriero and LeCun, 2024].

Why is ForestDiffusion better at vanilla generative modeling, while UnmaskingTrees is better on generation given partially-missing data and especially on imputation? We offer two speculative explanations. First, imputation is a conditional modeling scenario, except that you do not know the partition of the features into input features and output features *a priori*. One could address imputation by learning all possible 2^D conditional distributions, but this is impractical for large D, so one would prefer to learn a single joint distribution. Both autoregression and diffusion are ways of learning a joint distribution; because autoregression does so by learning conditional distributions, it is more suited to the conditional modeling imputation setting. Second, for missing data, diffusion has a train-inference gap: during training, observed features begin the reverse process from $\mathcal{N}(0, 1)$; during inference for imputation, observed features begin the reverse process at their actual values. On the other hand, the advantages of diffusion modeling (no quantization error, holistic generation) give it superiority when these problems can be avoided.

Despite their strong outperformance on other modalities, deep learning approaches have laboured against gradient-boosted decision trees on tabular data [Shwartz-Ziv and Armon, 2022, Jolicoeur-Martineau et al., 2024b]. Previous work [Breejen et al., 2024] suggests that tabular data requires an inductive prior that favors sharpness rather than smoothness, showing that TabPFN [Hollmann et al., 2022] (the leading deep learning tabular classification method) can be further improved with synthetic data generated from random forests. We anticipate that our XGBoost classifiers may be swapped out for a future variant of TabPFN that learns sharper boundaries and handles missingness.

We also note that MissForest [Stekhoven and Bühlmann, 2012], hailing from statistical literature on multiple imputation, has yet to be fully dethroned. Future progress in tabular conditional generation may require going back to the well of this traditional literature. As one example, we observe that MissForest exploits feature missingness fraction information, but we are not aware of any "machine learning" approaches which do so. The statistical literature has also previously explored the value of conditional modeling for joint modeling [Gelman and Raghunathan, 2001, Liu et al., 2014, Kropko et al., 2014]. Indeed, our UnmaskingTrees approach, and all autoregressive modeling, is presaged by the full-mechanism bootstrap [Efron, 1994].

Finally, we observe where randomness enters into our generation process. Flow-matching injects randomness solely at the beginning of the reverse process via Gaussian sampling, whereas diffusion models inject randomness both at the beginning and during the reverse process. In contrast, because our method starts with a fully-masked sample, it injects randomness gradually during the generation process. First, we randomly generate the order over features for unmasking. Second, we do not "greedily decode" to the most likely leaf in the meta-tree, but instead sample according to predicted probabilities. Third, for continuous features, having sampled a particular meta-tree leaf bin, we sample from within the bin, treating it as a uniform distribution.

5 Conclusions

We show that tree-based autoregressive unmasking is a strong, simple baseline for tabular data. We recommend UnmaskingTrees for tabular imputation, especially when downstream predictions are the goal, and for generation on datasets with missingness. We also offer BaltoBot as a fast, flexible method for computing predictive distributions.

References

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

Randall Balestriero and Yann LeCun. Learning by reconstruction produces uninformative features for perception. *arXiv preprint arXiv:2402.11337*, 2024.

Nicolas Beltran-Velez, Alessandro Antonio Grande, Achille Nazaret, Alp Kucukelbir, and David Blei. Treeffuser: Probabilistic predictions via conditional diffusions with gradient-boosted trees. *arXiv preprint arXiv:2406.07658*, 2024.

Felix den Breejen, Sangmin Bae, Stephen Cha, and Se-Young Yun. Why in-context learning transformers are tabular data classifiers. *arXiv preprint arXiv:2405.13396*, 2024.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Sander Dieleman. Diffusion is spectral autoregression, 2024. URL `https://sander.ai/2024/09/02/spectral-autoregression.html`.

Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In *International conference on machine learning*, pages 2690–2700. PMLR, 2020.

Bradley Efron. Missing data, imputation, and the bootstrap. *Journal of the American Statistical Association*, 89(426):463–475, 1994.

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

Andrew Gelman and Trivellore E Raghunathan. Using conditional distributions for missing-data imputation. *Statistical Science*, 15:268–69, 2001.

Manbir Gulati and Paul Roysdon. Tabmt: Generating tabular data with masked transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

David P Hofmeyr. Fast exact evaluation of univariate kernel sums. *IEEE transactions on pattern analysis and machine intelligence*, 43(2):447–458, 2019.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

Alexia Jolicoeur-Martineau, Aristide Baratin, Kisoo Kwon, Boris Knyazev, and Yan Zhang. Any-property-conditional molecule generation with self-criticism using spanning trees. *arXiv preprint arXiv:2407.09357*, 2024a.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In *International Conference on Artificial Intelligence and Statistics*, pages 1288–1296. PMLR, 2024b. URL `https://github.com/SamsungSAILMontreal/ForestDiffusion`.

Ouail Kitouni, Niklas Nolte, Diane Bouchacourt, Adina Williams, Mike Rabbat, and Mark Ibrahim. The factorization curse: Which tokens you predict underlie the reversal curse and more. *arXiv preprint arXiv:2406.05183*, 2024.

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

Jonathan Kropko, Ben Goodrich, Andrew Gelman, and Jennifer Hill. Multiple imputation for continuous and categorical data: comparing joint multivariate normal and conditional approaches. *Political Analysis*, 22(4), 2014.

Quoc V Le, Tim Sears, and Alexander J Smola. Nonparametric quantile regression. Technical report, Technical report, National ICT Australia, June 2005. Available at http://sml . . . , 2005.

Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11983–11992, 2020.

Jingchen Liu, Andrew Gelman, Jennifer Hill, Yu-Sung Su, and Jonathan Kropko. On the stationary distribution of iterative imputations. *Biometrika*, 101(1):155–173, 2014.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, June 2022.

Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. Tabpfgen–tabular data generation with tabpfn. *arXiv preprint arXiv:2406.05216*, 2024.

Alexander März. Xgboostlss–an extension of xgboost to probabilistic forecasting. *arXiv preprint arXiv:1907.03178*, 2019.

Calvin McCarter. The kernel density integral transformation. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36, 2024.

Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of machine learning research*, 7(6), 2006.

Severi Rissanen, Markus Heinonen, and Arno Solin. Generative modelling with inverse heat dissipation. *arXiv preprint arXiv:2206.13397*, 2022.

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

Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. Probabilistic gradient boosting machines for large-scale probabilistic regression. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 1510–1520, 2021.

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

Riley Stews. trasformers are kiki, diffusion is bouba, and language is pointier than images, 2024. URL `https://x.com/riley_stews/status/1827089629369266492`.

Stef Van Buuren, Hendriek C Boshuizen, and Dick L Knook. Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in medicine*, 18(6):681–694, 1999.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

Samuel Von Wilson, Bogdan Cebere, James Myatt, and Samuel Wilson. AnotherSamWilson/miceforest: Release for Zenodo DOI, December 2022. URL https://doi.org/10.5281/zenodo.7428632.

Z Yang. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Shuhan Zheng and Nontawat Charoenphakdee. Diffusion models for missing value imputation in tabular data. *arXiv preprint arXiv:2210.17128*, 2022.

A Ablation experiments with imputation

We additionally run UnmaskingTrees without BaltoBot, and instead with vanilla quantization using k-Means [Lloyd, 1982] and KDI [McCarter, 2023]. Average ranks (shown in Table 4) and raw scores (shown in Table 5) demonstrate that UnmaskingTrees on its own improves upon Forest-VP's diffusion approach. We also see that KDI quantization contributes to improvement beyond k-Means, and that BaltoBot yields even further improvement.

Table 4: Averaged ranks from ablation study of tabular data imputation (27 datasets, 3 experiments per dataset, 10 imputations per experiment) with 20% missing. Shown are *averaged rank* over all datasets and experiments (standard-error). Overall best is highlighted; better of Forest-VP versus ours is **boldface blue**. See Table 1 for column meanings.

	MinMAE ↓	AvgMAE ↓	W_{train} ↓	W_{test} ↓	MAD ↓	R^2 ↓	F_1 ↓	P_{bias} ↓	Cov_{rate} ↓
KNN	6.8 (0.6)	7.8 (0.6)	6 (0.4)	6.1 (0.5)	10.4 (0)	8.2 (1.3)	7 (1.5)	7.5 (1.5)	6.5 (0.8)
ICE	8.3 (0.5)	5.8 (0.5)	8.5 (0.6)	8.8 (0.5)	1.9 (0.4)	8 (1.1)	9 (0.6)	7.2 (1.1)	6.4 (0.8)
MICE-Forest	4.8 (0.6)	3.3 (0.6)	3.5 (0.3)	3.4 (0.3)	4.6 (0.4)	4.3 (1.8)	4.3 (1.3)	6.8 (1.6)	4.8 (0.7)
MissForest	3.3 (0.7)	5 (0.6)	2.2 (0.4)	2.3 (0.4)	7.2 (0.3)	4.7 (1.8)	3.3 (0.9)	6.8 (1.9)	3.8 (0.6)
Softimpute	8.3 (0.5)	9.3 (0.5)	8.8 (0.6)	8.9 (0.6)	10.4 (0)	7.5 (1.2)	9.8 (0.4)	8.3 (0.9)	7.9 (0.6)
OT	7.2 (0.5)	7.6 (0.4)	7.4 (0.6)	7.4 (0.6)	4.8 (0.4)	8.2 (0.5)	8.8 (0.6)	7.3 (0.7)	5.8 (0.7)
GAIN	5.8 (0.5)	8.3 (0.4)	7.2 (0.5)	7.5 (0.4)	8.9 (0.1)	7.5 (0.8)	7.4 (0.8)	6.7 (1)	6.1 (0.8)
Forest-VP	6.4 (0.5)	4.8 (0.6)	7 (0.4)	6.1 (0.5)	**3.8** (0.5)	6.5 (0.9)	6.6 (0.8)	4.5 (0.8)	6.5 (0.8)
UTrees-kMeans	6 (0.6)	5.8 (0.5)	6.3 (0.6)	6.1 (0.6)	4.1 (0.3)	4 (0.7)	**2.9** (0.6)	3.8 (1)	6 (0.7)
UTrees-KDI	5.1 (0.5)	5.1 (0.5)	5.4 (0.6)	5.6 (0.5)	4.8 (0.3)	4.5 (0.9)	4 (0.5)	**3.5** (1.2)	6.4 (0.7)
UTrees	**3.8** (0.5)	**3.2** (0.5)	**3.8** (0.4)	**3.8** (0.5)	5 (0.3)	**2.7** (0.6)	**2.9** (0.8)	**3.5** (0.8)	**5.8** (0.7)

Table 5: Raw scores from ablation study for tabular data imputation (27 datasets, 3 experiments per dataset, 10 imputations per experiment) with 20% missing values. Shown are raw scores - mean (standard-error). Overall best is highlighted; better of Forest-VP versus ours is **boldface blue**. See Table 1 for column meanings.

	MinMAE ↓	AvgMAE ↓	W_{train} ↓	W_{test} ↓	MAD ↑	R^2_{imp} ↑	$F1_{imp}$ ↑	P_{bias} ↓	Cov_{rate} ↑
KNN	0.16 (0.03)	0.16 (0.03)	0.42 (0.08)	1.89 (0.49)	0 (0)	0.59 (0.09)	0.75 (0.04)	1.27 (0.25)	0.4 (0.11)
ICE	0.1 (0.01)	0.21 (0.03)	0.52 (0.09)	1.99 (0.49)	0.69 (0.1)	0.59 (0.09)	0.74 (0.04)	1.05 (0.29)	0.39 (0.09)
MICE-Forest	0.08 (0.02)	0.13 (0.03)	0.34 (0.07)	1.86 (0.48)	0.29 (0.08)	0.61 (0.1)	0.76 (0.04)	0.61 (0.2)	0.75 (0.11)
MissForest	0.1 (0.03)	0.12 (0.03)	0.32 (0.07)	1.85 (0.48)	0.1 (0.03)	0.61 (0.1)	0.76 (0.04)	0.62 (0.22)	0.79 (0.08)
Softimpute	0.22 (0.03)	0.22 (0.03)	0.53 (0.07)	1.99 (0.48)	0 (0)	0.58 (0.09)	0.74 (0.04)	1.18 (0.34)	0.31 (0.09)
OT	0.14 (0.02)	0.19 (0.03)	0.56 (0.1)	1.98 (0.49)	0.28 (0.05)	0.59 (0.1)	0.75 (0.04)	1.09 (0.27)	0.39 (0.12)
GAIN	0.16 (0.03)	0.17 (0.03)	0.49 (0.11)	1.95 (0.51)	0.01 (0)	0.6 (0.1)	0.75 (0.04)	1.04 (0.25)	0.54 (0.12)
Forest-VP	0.14 (0.04)	0.17 (0.03)	0.55 (0.13)	1.96 (0.5)	0.25 (0.03)	**0.61** (0.1)	0.74 (0.04)	0.81 (0.25)	0.57 (0.14)
UTrees-kMeans	0.1 (0.02)	0.15 (0.03)	0.43 (0.09)	1.9 (0.5)	**0.28** (0.06)	**0.61** (0.1)	**0.76** (0.04)	0.63 (0.21)	**0.72** (0.13)
Utrees-KDI	0.1 (0.02)	**0.14** (0.03)	0.42 (0.09)	1.89 (0.49)	0.27 (0.06)	**0.61** (0.1)	**0.76** (0.04)	0.68 (0.24)	0.68 (0.14)
UTrees	**0.08** (0.02)	**0.14** (0.03)	**0.37** (0.08)	**1.87** (0.48)	0.27 (0.07)	**0.61** (0.1)	**0.76** (0.04)	**0.55** (0.19)	0.71 (0.13)
Oracle	0 (0)	0 (0)	0 (0)	1.87 (0.49)	0 (0)	0.64 (0.09)	0.78 (0.04)	0 (0)	1 (0)