Generating and Imputing Tabular Data via Diffusion and Flow-based Gradient-Boosted Trees

Alexia Jolicoeur-Martineau
Samsung - SAIT AI Lab

Kilian Fatras†
Mila & McGill University

Tal Kachman
Radboud University & Donders Institute

Abstract

Tabular data is hard to acquire and is subject to missing values. This paper introduces a novel approach for generating and imputing mixed-type (continuous and categorical) tabular data utilizing score-based diffusion and conditional flow matching. In contrast to prior methods that rely on neural networks to learn the score function or the vector field, we adopt XGBoost, a widely used Gradient-Boosted Tree (GBT) technique. To test our method, we build one of the most extensive benchmarks for tabular data generation and imputation, containing 27 diverse datasets and 9 metrics. Through empirical evaluation across the benchmark, we demonstrate that our approach outperforms deep-learning generation methods in data generation tasks and remains competitive in data imputation. Notably, it can be trained in parallel using CPUs without requiring a GPU. Our Python and R code is available at `https://github.com/SamsungSAILMontreal/ForestDiffusion`.

1 INTRODUCTION

Tabular datasets are omnipresent across various fields like economics, medicine, and social sciences. Two interconnected fundamental challenges with this type of data are small training datasets and missing values(Little and Rubin, 1989; Bennett, 2001). To make the problems worse, different participants may have missing data for different variables. Consequently, training a model solely on complete cases would significantly shrink the participant pool, potentially exacerbating the already limited dataset size (Muzellec et al., 2020), thereby reducing statistical power and leading to bias (Donner, 1982).

The most common method to deal with missing data is to use single or multiple imputations. Single imputation replaces the missing values in the dataset with predictions and trains the model on this imputed dataset. This can work for prediction or classification, but relying on a single guess for each missing data point can lead to incorrect inferences because of the bias in the imputed dataset. Multiple imputations generate several imputed datasets, each with its own set of imputed values. Then, a separate model is trained per imputed dataset, and the results are combined to account for the uncertainty inherent in the imputation process (Little and Rubin, 1987).

Remarkably, there has been significant advancements in the field of generative models such as images (Brock et al., 2018; Karras et al., 2019; Rombach et al., 2022), audio (Chen et al., 2020; Kong et al., 2020), videos (Voleti et al., 2022; Harvey et al., 2022), graphs (Niu et al., 2020), and tabular data (Kim et al., 2022; Kotelnikov et al., 2023; Borisov et al., 2022b). These methods offer the possibility to artificially expand datasets, addressing the challenge of limited data akin to data augmentation techniques Mumuni and Mumuni (2022). Furthermore, the latest generative models have demonstrated remarkable effectiveness at inpainting (Meng et al., 2021; Lugmayr et al., 2022; Zheng and Charoenphakdee, 2022; Ouyang et al., 2023), which involves restoring missing image portions, much like imputing missing data in the case of tabular data (Yun et al., 2023). This demonstrates the potential of generative models to alleviate the challenges of limited data and missing values encountered in tabular datasets.

One of the most successful recent generative models is Diffusion Models (DMs) (Sohl-Dickstein et al., 2015; Song and Ermon, 2019, 2020; Song et al., 2021; Ho et al., 2020). DMs estimate the score-function (gradient log density) and rely on Stochastic Differential Equations (SDEs) to generate samples. A more recent approach called Conditional Flow Matching (CFM) instead estimates a vector field and relies on Ordinary

† Now at Dreamfold. Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

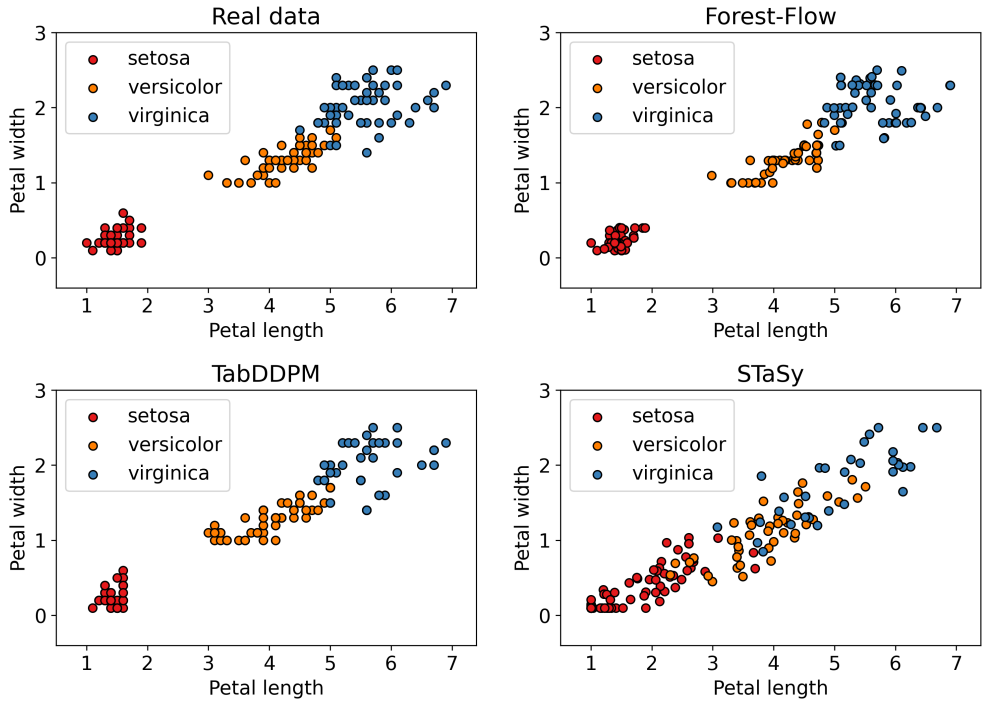


Figure 1: Iris dataset: Three-way interaction between Petal length, width, and species using real or fake samples (using Forest-Flow, our XGBoost method, or TabDDPM and STaSy, deep-learning diffusion methods)

Differential Equations (ODEs) to generate data (Liu et al., 2022; Albergo and Vanden-Eijnden, 2023; Lipman et al., 2022; Tong et al., 2023b,a). Both methods have been successful in data generation tasks.

Traditionally, DMs and CFMs have relied on deep neural networks to estimate the score-function or the vector field because Neural Networks (NNs) are considered Universal Function Approximators (UFAs) (Hornik et al., 1989). However, many other UFAs exist. For example, Decision Trees and more complex tree-based methods such as Random Forests (Breiman, 2001) or Gradient-Boosted Trees (GBTs) (Friedman et al., 2000; Friedman, 2001) are also UFAs (Royden and Fitzpatrick, 1968; Watt et al., 2020). Furthermore, for tabular data prediction and classification, GBTs tend to perform better than neural networks (Shwartz-Ziv and Armon, 2022; Borisov et al., 2022a; Grinsztajn et al., 2022), and most of them can natively handle missing data through careful splitting (Chen et al., 2015; Prokhorenkova et al., 2018).

Because Gradient-Boosted Trees (GBTs) perform particularly well on tabular data, we sought to use GBTs as function estimators in flow and diffusion models. In doing so, our **main contributions** are:

- We create the first diffusion and flow models for tabular data generation and imputation using XGBoost (see Figure 1), a popular GBT method, instead of neural networks.
- Contrary to most generative models for tabular data, our method can be trained directly on incomplete data thanks to XGBoost, which learns the best split for missing values.
- We provide an extensive benchmark for generation and imputation methods on 27 real-world datasets with a wide range of evaluation metrics tackling four quadrants: closeness in distribution, diversity, prediction, and statistical inference.
- Our method generates highly realistic synthetic data when the training dataset is either clean or tainted by missing data.

2 BACKGROUND

2.1 Gradient-boosted trees and XGBoost

Decision Trees are predictive models that partition input data into distinct subsets via decision splits, thereby culminating in terminal nodes, each furnishing a definitive prediction. They recursively partition the feature space to maximize the homogeneity of predictions within each partition. By strategically selecting decision splits based on certain criteria, they efficiently maximize predictive performance.

GBTs (Friedman et al., 2000; Friedman, 2001) take decision trees a step further by building decision trees

sequentially, where each tree corrects the errors made by the previous one. It starts with a simple tree, often called a weak learner, and then iteratively adds more trees while emphasizing the examples that the previous trees predicted incorrectly. This iterative process continues until a specified number of trees are built or until a certain level of accuracy is achieved. GBTs have shown great success for tabular data prediction and classification (Zhang et al., 2017; Touzani et al., 2018; Machado et al., 2019; Ma et al., 2020).

XGBoost (eXtreme Gradient Boosting) is a popular open-source GBT. It uses a second-order Taylor expansion, fast parallelized/distributed system, fast quantile splitting, and sparsity-aware split finding (allowing it to naturally handle missing values) for maximum performance (Shwartz-Ziv and Armon, 2022; Florek and Zagdański, 2023). We rely on XGBoost as our ablation (see Section 4.4) showed it performed the best.

2.2 Generative diffusion and conditional flow matching models

Previous generations of generative models, such as GANs Goodfellow et al. (2014) or VAEs Kingma and Welling (2013), requires to differentiate through two models. This prevents the use of non-differentiable models such as GBTs. However, the new diffusion and flow-based models only need one model, which paves the way for using GBTs in generative modeling.

SDEs and score-based models The purpose of generative models is to generate realistic data from Gaussian noise. As highlighted in Song et al. (2021), a possible manner to generate data can be done through stochastic differential equations (SDE) (Feller, 1949). The forward diffusion process consists of transforming data into Gaussian noise through an SDE of the form:

$$dx = u_t(x)\,dt + g(t)dw, \tag{1}$$

where $t \in [0,1]$, $u : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is a smooth time-varying vector field, $g : \mathbb{R} \to \mathbb{R}$ is a scalar function and w is a Brownian motion. Different choices of forward diffusion process exists to ensure that $x(t=0)$ is a real data, while $x(t=1)$ is pure Gaussian noise. Some of the popular choices are the Variance Preserving (VP) (Ho et al., 2020) or the Variance Exploding (VE) (Song and Ermon, 2019) SDEs. Importantly, it is possible to reverse the forward diffusion process (1) through a reverse SDE of the form:

$$dx = [u_t(x) - g(t)^2 \nabla_x \log p_t(x)]\,dt + g(t)d\bar{w}, \tag{2}$$

where $\bar{w}$ is the reverse Brownian motion. The main idea of score-based models is to i) learn the score function $\nabla_x \log p_t(x)$ by perturbing real data with Gaussian noise Song and Ermon (2019) with the score-matching loss, which reads:

$$l_{\text{sm}}(\theta) = \mathbb{E}_t \lambda_t^2 \mathbb{E}_{x_0, x_t|x_0} \|s_\theta(t, x_t) - \nabla \log p_t(x_t|x_0)\|^2, \tag{3}$$

where $t \sim U(0,1)$ and λ_t is a positive weight. Then, ii) generate fake data by solving the reverse SDE (2) with the score function approximation while starting at random Gaussian noise (effectively going backward from noise to data). In practice, to solve the reverse SDE, we generally discretize t over a fixed set of noise levels n_t going from $t=1$ to $t=0$.

ODEs and conditional flow matching When $g(t) = 0$ in (1), we recover the setting of an ordinary differential equation (ODE), making the forward and reverse flow deterministic and simplifying the problem to be solved. We denote by $\phi_t(x)$ the solution of this ODE with initial conditions $\phi_0(x) = x$; that is, $\phi_t(x)$ is the point x transported along the vector field u from $t=1$ up to $t=0$. Given a density p_0 over $\mathbb{R}^d$, we can define the time-varying density $p_t = [\phi_t]_\#(p_0)$, where $[\phi_t]_\#$ is the pushforward operator of ϕ. By construction, the density p_t verifies the well-known *continuity equation* with initial conditions p_0.

Approximating ODEs with neural networks is a rich research direction (Kidger, 2022). Chen et al. (2018) approximated the vector field $u_t(x)$ with a neural network $v_\theta(t,x) : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$. However, this direction requires the simulation of the ODE, which does not scale well to high dimensions. Another direction is to regress $v_\theta(t,x)$ against the vector field $u_t(x)$. Unfortunately, this loss is intractable as it requires knowing the probability path $p_t(x)$ and the vector field $u_t(x)$.

A workaround was proposed in Lipman et al. (2022). In their framework, they assume that the probability path is a mixture of conditional probability paths. Formally, $p_t(x) = \int p_t(x|z)q(z)dz$, where q is a latent distribution and the conditional probability paths $p_t(x|z)$ are supposed to be generated from some conditional vector fields $\mu_t(x|z)$. Then, they defined the following vector field: $\mu_t(x) = \mathbb{E}_{q(z)} \frac{\mu_t(x|z)p_t(x|z)}{p_t(x)}$ and showed that it generates the probability path $p_t(x)$ (*i.e.,* μ_t and p_t verify the continuity equation). They also proved that regressing $v_\theta(t,x)$ against the conditional vector field $\mu_t(x|z)$ leads to the same gradients as regressing $v_\theta(t,x)$ against the vector field $\mu_t(x)$. Therefore, they minimize:

$$l_{\text{cfm}}(\theta) = \mathbb{E}_{t, q(z), p_t(x|z)} \|v_\theta(t,x) - \mu_t(x|z)\|^2. \tag{4}$$

After training v_θ, they generate fake samples by solving the following ODE ($t=1$ to $t=0$)[1]:

$$dx = v_\theta(t,x)\,dt. \tag{5}$$

[1]Note that we have considered ODE from $t=1$ to $t=0$ for consistencies with respect to diffusion models.

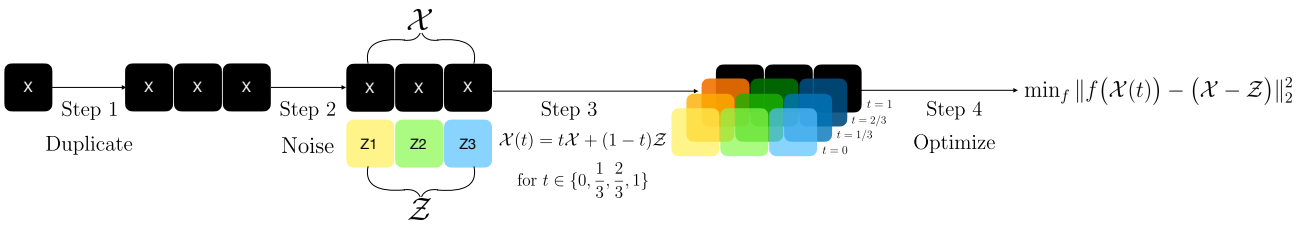


Figure 2: Illustration of our Forest-Flow method based on I-CFM Tong et al. (2023b) (see §A.3 for more details). The first step duplicates the original dataset. The second step adds a different noise to each duplicated dataset. The third step computes the linear interpolation between the duplicated dataset and their corresponding noise for different time t (*i.e.,* $\mathcal{X}_i(t) = t\mathcal{X} + (1-t)\mathcal{Z}_i, \forall i \in [1, \ldots, n_{noise}]$ and $\forall t \in t_{\text{levels}}$). The final step is to regress a GBT model at each noise level against the vector field; the training of the n_t models is parallelized over CPUs.

This direction has made ODE-based generative models competitive compared to their stochastic variant.

In practice, we use the I-CFM method Tong et al. (2023b). This method assumes that z is a tuple of noise and real data (*i.e.,* $z = (x_0, x_1)$) and the distribution q is the independent coupling $q(x_0, x_1) = q(x_0)q(x_1)$. We choose the conditional probability path to be $p_t(x|(x_0, x_1)) = \mathcal{N}(tx_1 + (1-t)x_0, \sigma)$ for $\sigma > 0$. This results in the following conditional vector field: $\mu_t(x|(x_0, x_1)) = x_1 - x_0$. We refer to §A.3 for a longer discussion on Flow Matching.

3 TRAINING DIFFUSION & FLOW MODELS WITH XGBOOST

Traditionally (and exclusively as far as we know), flow and diffusion models have relied on deep neural networks. Instead, our method relies on Gradient-boosted Trees (GBTs) to estimate the vector field or score-function. We explain the details of our method below (see also Figure 2 for an illustration of the method).

Let us assume that we have a training dataset X of size $[n, d]$ and that we have discrete set of n_t noise levels $t \in t_{levels} = \{\frac{1}{n_t}, \ldots, \frac{n_t - 1}{n_t}, 1\}$, where $t = 0$ corresponds to real data, while $t = 1$ corresponds to Gaussian noise.

3.1 Duplicating the dataset to estimate the expectation in diffusion and flow losses

A key component of NNs training is Stochastic Gradient Descent (SGD). To use SGD for training a diffusion model, a mini-batch of data of size b is selected from the training dataset along with random Gaussian noise of the same size (i.e., $[b, d]$). For each of the b samples, a random noise level t and a Gaussian noise z (of size d) are sampled. These are then used to compute the noisy sample $x(t)$ using the forward diffusion/flow step. By using SGD with random sampling, we can minimize the expectations (3) and (4) over mini-batch of data, Gaussian noise, and noise level.

Unfortunately, GBTs do not rely on SGD and are instead trained on the entire dataset (although each tree can be trained on different subsamples of the data as a regularization). Therefore, we need to prepare a single input and output training dataset precomputed in advance to approximate the expectations (3) and (4). Since the expectations are over all possible noise-data pairs, we need to sample multiple Gaussian noise z per data sample x, *i.e.,* we build a set $\{(z_i, x)_{i \in [1, \ldots, n_{noise}]}\}$. To achieve this, we duplicate the observations (*i.e.,* the rows) of the original dataset n_{noise} times (going from size $[n, d]$ to size $[n_{noise}n, d]$) as illustrated in Step 1 of Figure 2. Then, we sample random Gaussian noise of the same dimension ($[n_{noise}n, d]$), as illustrated in Step 2 of Figure 2, so that each sample get n_{noise} different random noise. These two datasets (containing the duplicated data samples and the random Gaussian noise) are used to produce one fixed dataset $X(t)$ containing all the pre-calculated noisy samples $x(t)$ and one fixed dataset $Y(t)$ containing their corresponding outputs $y(t)$ ($\nabla \log p_t(x_t|x_0)$ for diffusion model and $\mu_t(x|z)$ for flow model) for all t[2]. It is illustrated in the third step of Figure 2. The entire process for our Flow-based method is illustrated in Figure 2.

In practice, n_{noise} is one of the most critical hyper-parameters for achieving good performance because it controls the approximation of the expectation over data-noise pairs. Therefore, n_{noise} should be as high as possible, given memory constraints. We set $n_{noise} = 100$ for all datasets, unless memory is an issue (when nd is large), in which case we reduce it to $n_{noise} = 50$.

3.2 Training different models per noise level

As seen from (3) and (4), the model depends on both x_t and t. Since training a neural network to approximate

[2]Since the release of XGBoost 2.0, data duplication can be avoided using XGBoost's novel data iterator which mimics minibatches. Other GBTs still require this procedure.

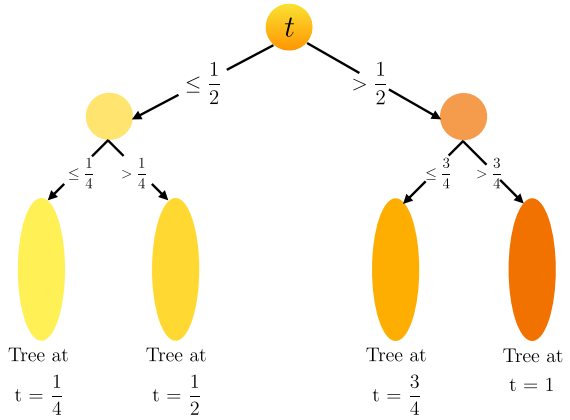


Figure 3: Our method learns a different GBT model (with 100 trees) for each noise-level (here $n_t = 4$). We can re-interpret this as a single model with giants trees where the time variable splits are hard-coded.

the vector field or score-function can be very slow, it is usual to take the noise level t as an input (which is processed through a complex Sinusoidal (Vaswani et al., 2017) or Fourier feature (Tancik et al., 2020) embedding). The same idea can be done with GBTs by taking t as an additional model feature. However, this approach is problematic when the number of variables d is large. For simplicity, assume that the variables used for the tree splits are randomly chosen, then the probability of splitting a node by t is $\frac{1}{d+1}$. This highlights the fact that with many variables, the chance of splitting by the noise level is very small, which greatly minimizes the influence of t on the output.

To better differentiate the vector field or score at different noise levels, *we train a different model per noise level* t, leading to n_t models whose training can be parallelized over multiple CPUs. cost. It can be seen as a model where the time variable splits are hard-coded (see Figure 3). In practice, $n_t = 50$ is enough to reach state-of-the-art or competitive performance. Increasing n_t could lead to greater performance but at the price of a higher computational cost.

3.3 Choice of Gradient-boosted Trees

In theory, we could use any Gradient-boosted Trees. We experimented our method with Random Forests (Ho, 1995; Breiman, 2001), XGBoost (Chen et al., 2015), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). We have found that XGBoost achieved superior performance than all other GBT methods (see Ablation §4.4). Furthermore, Random Forests cannot be trained with missing data, motivating our choice to exclusively use XGBoost as GBT for our approach (see §3.5).

Algorithm 1: Forest-Diffusion Training

Input: Dataset X of size $[n, d]$, $n_{noise} = 100$ noise sample per data sample, $n_t = 50$ noise levels.
$X' \leftarrow$ Duplicate the rows of X n_{noise} times
$Z \leftarrow$ Dataset of $z \sim \mathcal{N}(0, 1)$ with the size of X'
$t_{levels} = \{\frac{1}{n_t}, \dots, \frac{n_t - 1}{n_t}, 1\}$
`// Parallelized on CPUs`
for $t \in t_{levels}$ **do**
  $X(t), Y(t) \leftarrow \text{Forward}(Z, X', 0, t \mid n_t)$
  $f(t) \leftarrow$ Regression XGBoost model to predict $Y(t)$ given $X(t)$
end
return $\{f(t)\}_{t \in t_{levels}}$

Algorithm 2: Forest-Diffusion Sampling

Input: XGBoost models $\{f(t)\}_{t \in t_{levels}}$, n_{obs} samples, d variables, $n_t = 50$ noise levels.
$X(1) \leftarrow$ Dataset of $z \sim \mathcal{N}(0, 1)$ with size $[n_{obs}, d]$
$t \leftarrow 1$
while $t > 0$ **do**
  $X(t - \frac{1}{n_t}) \leftarrow \text{Reverse}(X(t) \mid f(t), t, n_t)$
  $t \leftarrow t - \frac{1}{n_t}$
end
return $X(0)$

3.4 Forward Diffusion/Flow process

There are many possible choices for the forward process of diffusion and flow models. For diffusion, we use the Variance Preserving (VP) process (Song et al., 2021). For CFM, we use the deterministic independent-coupling conditional flow matching (I-CFM) (Tong et al., 2023b). See Appendix A.2 for the forward and reverse processes of VP and I-CFM.

3.5 Imputation via diffusion and XGBoost

As previously mentioned, inpainting (restoring missing parts of an image) is effectively the same problem as imputing missing data when dealing with tabular data. For this reason, as a method of imputation, we use REPAINT (Lugmayr et al., 2022), a powerful inpainting diffusion models method.

A critical distinction between the training of diffusion models for image inpainting and the training of GBTs diffusion/flow models is that inpainting models are trained on complete images. In contrast, we must train on incomplete data. For this reason alone, XGBoost is particularly interesting for this task because it can handle missing data by learning the best splitting direction for missing values. Thus, we leverage this property to train a diffusion model on incomplete data

Algorithm 3: Forest-Diffusion Imputation

Input: XGBoost models $\{f(t)\}_{t\in t_{levels}}$, dataset X of size $[n_{obs}, d]$, $n_t = 50$ noise levels.
$M \leftarrow$ Mask indicating which values are non-missing in X (1: non-missing, 0: missing)
$X(1) \leftarrow$ Dataset of $z \sim \mathcal{N}(0,1)$ with size $[n_{obs}, d]$
$t \leftarrow 1$
while $t > 0$ **do**
  $X(t - \frac{1}{n_t}) \leftarrow$ Empty Dataset with size $[n_{obs}, d]$
  `// Reverse process for missing values`
  $X(t - \frac{1}{n_t})[1-M] \leftarrow \text{Reverse}(X(t)[1-M] \mid f(t), t, n_t)$
  `// Set non-missing values to truth`
  $X(t-\frac{1}{n_t})[M] \leftarrow \text{Forward}(Z[M], X[M], 0, t \mid n_t)$
  $t \leftarrow t - \frac{1}{n_t}$
end
return $X(0)$

and use it to impute missing data with REPAINT.

Importantly, REPAINT is an algorithm made for diffusion models. By their deterministic nature, flow models are not suited for inpainting and we are not aware of any CFM method which tackles inpainting. We provide some additional intuition as to why it may not be possible to impute nor inpaint in Appendix A.4.

3.6 Data processing

Pre-processing Tabular data can contain both numerical and categorical variables. Our method takes continuous variables in $\mathbb{R}$. We make the categorical variables continuous by dummy encoding them (Suits, 1957). Then, we min-max normalize every variable (including dummies) to the range $[-1, 1]$.

Post-processing After data generation (or imputation), we clip the generated values between $[-1, 1]$ and reverse the min-max normalization. For integer variables, we also round these variables to remove any decimals. Finally, we round the dummy variables to the nearest class to obtain the categorical variables.

3.7 Gradient-Boosted Tree hyperparameters

Since our goal is to estimate the flow or score-function of the distribution, overfitting is of very little concern, while underfitting is a major concern. For this reason, we do not use any regularization, such as L_1 or L_2 penalizations. Outside of the L_2 hyperparameter, which we set to 0, all other hyperparameters of XGBoost are left at their default values (note that the default number of trees is 100).

Table 1: Evaluation metrics used in our experiments

Metric	Abbreviation	Purpose
Distance in distribution (or to ground-truth)		
Wasserstein distance	W_{train}, W_{test}	Distance to train/test data distributions
Minimum Mean Absolute Error	MinMAE	Distance between closest imputation and non-missing data
Average Mean Absolute Error	AvgMAE	Average distance between imputation and non-missing data
Diversity		
Coverage	cov_{train}, cov_{test}	Diversity of fake samples relative to train/test data
Mean Absolute Deviation	MAD	Diversity of imputations
Prediction		
R-squared and F1-score	R^2_{fake}, $F1_{fake}$, R^2_{imp}, $F1_{imp}$	Usefulness of fake or imputed data for Machine Learning prediction/classification
Discriminator F1-score	$F1_{disc}$	Ability to distinguish real from fake data
Statistical inference		
Percent Bias	P_{bias}	Regression parameter precision
Coverage Rate	cov_{rate}	Regression parameter confidence intervals coverage

3.8 Training one model per category

When the dataset has a categorical outcome (i.e., a dataset for classification), Kotelnikov et al. (2023) train their score function conditional on the label. Then, they generate data by i) sampling a random label with the same probabilities as the training data and ii) generating fake data conditional on that label. They do so to improve performance and reduce the degrees of freedom by one. We use this idea, but instead of conditioning on the label, we train a different XGBoost model per label. It is equivalent to forcing each tree to split by the labels before building deep trees. We find that it improves generation performance.

3.9 Algorithm details

The training algorithm is described in Algorithm 1. The sampling and imputation methods are described in Algorithms 2 and 3. Note that we removed REPAINT from the imputation algorithm for simplicity, but the algorithm using REPAINT can be found in the Appendix (see Algorithm 4). Again, for simplicity, We also leave out the details on the Forward and Reverse process in the Appendix (see Algorithms 5, 6, and 7).

4 EXPERIMENTS

We evaluate our method on 27 real-world classification/regression datasets (with input X and outcome y) on the following tasks: 1) generation with complete data, 2) imputation, and 3) generation with in-

Table 2: Tabular data generation with complete data (27 datasets, 3 experiments per dataset); *averaged rank* over all datasets and experiments (standard-error). Best highlighted in **bold**.

	$W_{train}\downarrow$	$W_{test}\downarrow$	$cov_{train}\downarrow$	$cov_{test}\downarrow$	$R^2_{fake}\downarrow$	$F1_{fake}\downarrow$	$F1_{disc}\downarrow$	$P_{bias}\downarrow$	$cov_{rate}\downarrow$
GaussianCopula	6.1 (0.3)	6.2 (0.3)	6.3 (0.3)	6.4 (0.3)	5.2 (0.2)	5.6 (0.3)	6.1 (0.4)	5.7 (1.0)	6.7 (0.6)
TVAE	4.3 (0.2)	4.1 (0.2)	4.7 (0.2)	4.7 (0.2)	5.5 (0.7)	5.2 (0.5)	4.9 (0.2)	6.5 (0.5)	6.0 (0.4)
CTGAN	7.4 (0.1)	7.4 (0.2)	7.3 (0.2)	7.1 (0.2)	7.5 (0.2)	7.3 (0.2)	6.1 (0.3)	4.7 (1.0)	6.3 (0.5)
CTAB-GAN+	5.8 (0.3)	5.7 (0.3)	6.3 (0.3)	6.1 (0.3)	6.0 (0.3)	6.0 (0.3)	6.5 (0.2)	6.8 (0.7)	6.0 (0.8)
STaSy	5.1 (0.2)	5.3 (0.2)	4.3 (0.2)	4.4 (0.2)	5.5 (1.0)	4.3 (0.3)	5.1 (0.3)	4.2 (0.7)	3.9 (0.9)
TabDDPM	2.7 (0.6)	3.4 (0.5)	2.6 (0.4)	2.9 (0.4)	**1.2** (0.2)	3.4 (0.5)	**2.0** (0.4)	2.7 (0.7)	**1.4** (0.2)
Forest-VP	2.7 (0.1)	2.5 (0.1)	2.9 (0.2)	2.7 (0.3)	2.8 (0.3)	**2.0** (0.2)	2.6 (0.3)	3.0 (0.8)	3.2 (0.6)
Forest-Flow	**1.8** (0.1)	**1.4** (0.1)	**1.6** (0.2)	**1.6** (0.2)	2.3 (0.4)	2.1 (0.3)	2.7 (0.3)	**2.5** (0.3)	2.5 (0.3)

Table 3: Tabular data imputation (27 datasets, 3 experiments per dataset, 10 imputations per experiment) with 20% missing values; *averaged rank* over all datasets and experiments (standard-error). Best highlighted in **bold**.

	MinMAE $\downarrow$	AvgMAE $\downarrow$	$W_{train}\downarrow$	$W_{test}\downarrow$	MAD $\downarrow$	$R^2_{imp}\downarrow$	$F1_{imp}\downarrow$	$P_{bias}\downarrow$	$Cov_{rate}\downarrow$
KNN	4.8 (0.4)	5.5 (0.4)	4.2 (0.4)	4.2 (0.3)	7.4 (0.0)	5.7 (0.9)	5.0 (1.0)	5.5 (0.8)	4.8 (0.5)
ICE	6.0 (0.4)	3.9 (0.4)	6.2 (0.5)	6.4 (0.4)	**1.4** (0.2)	5.2 (1.0)	6.0 (0.6)	4.8 (0.8)	4.7 (0.6)
MICE-Forest	3.4 (0.4)	**2.0** (0.3)	2.5 (0.2)	2.5 (0.3)	3.2 (0.2)	**3.3** (1.2)	2.6 (0.9)	4.8 (1.0)	3.9 (0.6)
MissForest	**2.3** (0.4)	3.5 (0.4)	**1.5** (0.2)	**1.7** (0.3)	4.6 (0.1)	**3.3** (1.2)	**1.9** (0.4)	4.8 (1.3)	**3.0** (0.4)
Softimpute	5.9 (0.4)	6.7 (0.3)	6.2 (0.4)	6.5 (0.4)	7.4 (0.0)	5.2 (0.8)	6.8 (0.4)	5.3 (0.9)	5.9 (0.4)
OT	5.2 (0.4)	5.3 (0.3)	5.2 (0.4)	5.2 (0.4)	3.2 (0.2)	5.2 (0.5)	5.8 (0.6)	4.5 (0.8)	4.3 (0.5)
GAIN	3.9 (0.4)	5.6 (0.3)	5.2 (0.3)	5.2 (0.2)	5.9 (0.1)	4.7 (0.8)	4.4 (0.8)	3.7 (1.0)	4.5 (0.5)
Forest-VP	4.5 (0.4)	3.5 (0.4)	5.0 (0.3)	4.5 (0.4)	2.9 (0.3)	3.5 (0.9)	3.6 (0.8)	**2.5** (0.7)	4.8 (0.6)

complete data. For all tasks, we split the datasets in training (80%) and testing (20%) splits. We consider similar datasets as Muzellec et al. (2020), and they are from the UCI Machine Learning Repository (Dua and Graff, 2017) or scikit-learn (Pedregosa et al., 2011). The list of all datasets can be found in §B.1. To get a broad and careful assessment of the data quality, we consider a wide range of evaluation metrics accross four quadrants: closeness in distribution (or ground truth for imputations), diversity, prediction, and statistical inference (comparing regression parameters estimated with the true data versus those estimated with fake/imputed data). We summarize the evaluation metrics in Table 1. For more details on the choice and definitions of metrics, we refer to §B.2.

To condense the information across the 27 datasets, we report the **average rank (with standard-error)** of each method relative to other methods. Looking at the average rank allows for easier interpretation and reduces outliers by preventing a single low or high performance from providing an unfair (dis)advantage. We provide the tables with the **averaged raw scores** of each evaluation metric as well as the **bar plots** for each dataset, metric and method in §C.1 and §C.2. We denote our method **Forest-VP** with VP-diffusion and **Forest-Flow** with conditional flow matching.

4.1 Generation with complete tabular data

For the generation task, we generate both input X and outcome y. We compare our method to a wide range of tabular data generative models. We consider a statistical method: Gaussian Copula (Joe, 2014; Patki et al., 2016). We also consider deep-learning VAE and GAN based methods: TVAE (Xu et al., 2019), CTGAN (Xu et al., 2019), CTAB-GAN+ (Zhao et al., 2021). Finally, we consider deep-learning diffusion methods: STaSy (Kim et al., 2022), and TabDDPM (Kotelnikov et al., 2023). Details on methods can be found in §B.3.

Averaged score of the different methods are presented in Table 2, their raw scores in Table 8 and the bar plots in C.2.1. We find that Forest-Flow is the highest performing method (across nearly all metrics), followed closely by both Forest-VP and TabDDPM. Forest-Flow even nearly match the Wasserstein distance to the test set of the training data (denoted oracle in the table). This shows that our method does performs extremely well at generating realistic synthetic data.

4.2 Imputation on missing tabular data

For imputation, we randomly remove 20% of the values in the input X (Missing Completely at Random; MCAR). The outcome y is left as non-missing. We compare our method to a wide range of tabular data generative models. We consider non-deep methods: kNN-Imputation (Troyanskaya et al., 2001), ICE (Van Buuren and Groothuis-Oudshoorn, 2011; Buck, 1960), MissForest (Stekhoven and Bühlmann, 2012), MICE-Forest (Van Buuren et al., 1999; Wilson et al., 2023), softimpute (Hastie et al., 2015) and minibatch Sinkhorn divergence (Muzellec et al., 2020). We also consider deep-learning methods: MIDAS (Lall and Robinson, 2022), and GAIN (Yoon et al., 2018). Details on each method can be found in Appendix B.3.

Table 4: 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 and Forest-Flow; *average rank* (standard-error) over all datasets and experiments. Best highlighted in **bold**.

	$W_{train} \downarrow$	$W_{test} \downarrow$	$cov_{train} \downarrow$	$cov_{test} \downarrow$	$R^2_{fake} \downarrow$	$F1_{fake} \downarrow$	$F1_{disc} \downarrow$	$P_{bias} \downarrow$	$cov_{rate} \downarrow$
GaussianCopula	6.0 (0.3)	6.2 (0.2)	6.3 (0.3)	6.1 (0.3)	5.3 (0.4)	5.8 (0.2)	6.4 (0.5)	4.7 (0.8)	6.7 (0.6)
TVAE	4.2 (0.3)	3.9 (0.2)	4.8 (0.3)	4.8 (0.2)	5.0 (1.0)	4.9 (0.5)	5.0 (0.3)	7.0 (0.4)	5.5 (0.8)
CTGAN	7.3 (0.2)	7.4 (0.2)	7.4 (0.2)	7.3 (0.2)	7.3 (0.3)	7.4 (0.2)	6.0 (0.2)	4.2 (1.1)	6.1 (0.7)
CTABGAN	5.7 (0.4)	5.6 (0.3)	6.1 (0.3)	5.8 (0.3)	6.5 (0.4)	6.3 (0.3)	6.1 (0.3)	6.7 (0.8)	5.2 (0.5)
Stasy	4.9 (0.2)	5.1 (0.3)	4.3 (0.2)	4.1 (0.3)	5.2 (0.7)	3.7 (0.3)	4.3 (0.3)	3.3 (0.4)	4.2 (0.9)
TabDDPM	2.7 (0.6)	3.0 (0.6)	**2.1** (0.4)	2.7 (0.5)	**1.3** (0.2)	2.9 (0.5)	**2.0** (0.4)	3.3 (1.1)	**1.7** (0.3)
Forest-VP	2.8 (0.2)	2.7 (0.2)	3.0 (0.2)	3.0 (0.3)	2.7 (0.2)	**1.9** (0.2)	2.6 (0.3)	3.7 (0.7)	3.8 (1.0)
Forest-Flow	**2.3** (0.3)	**2.2** (0.3)	**2.1** (0.2)	**2.2** (0.2)	2.7 (0.6)	3.2 (0.3)	3.7 (0.3)	**3.2** (0.7)	2.8 (0.7)

Table 5: Forest-Flow ablation on the Iris dataset (baseline: $n_t = 50$, $n_{noise} = 100$, $y_{cond} = False$)

	$W_{test} \downarrow$	$cov_{test} \uparrow$	$F1_{fake} \uparrow$
$n_t = 10$	0.34	0.93	0.92
$n_t = 25$	0.32	0.93	0.95
(Base) $n_t = 50$	0.32	0.92	0.95
$n_t = 100$	0.33	0.92	0.93
$n_t = 200$	0.33	0.95	0.94
$n_{noise} = 1$	0.52	0.86	0.81
$n_{noise} = 5$	0.38	0.91	0.90
$n_{noise} = 10$	0.37	0.91	0.93
$n_{noise} = 25$	0.36	0.93	0.92
$n_{noise} = 50$	0.35	0.93	0.91
(Base) $n_{noise} = 100$	0.32	0.92	0.95
(Base) XGBoost	0.32	0.92	0.95
LightGBM	0.32	0.94	0.93
CatBoost	0.48	0.78	0.85
Random Forests	0.55	0.78	0.91
(Base) $y_{cond} = False$	0.34	0.92	0.95
$y_{cond} = True$	0.35	0.92	0.97

Table 6: Forest-VP ablation on the Iris dataset (baseline: $n_t = 50$, $n_{noise} = 100$, $y_{cond} = False$)

	$W_{test} \downarrow$	$cov_{test} \uparrow$	$F1_{fake} \uparrow$
$n_t = 10$	0.57	0.56	0.84
$n_t = 25$	0.37	0.88	0.93
(Base) $n_t = 50$	0.33	0.93	0.96
$n_t = 100$	0.34	0.93	0.95
$n_t = 200$	0.35	0.91	0.95
$n_{noise} = 1$	0.65	0.71	0.69
$n_{noise} = 5$	0.44	0.89	0.90
$n_{noise} = 10$	0.40	0.90	0.93
$n_{noise} = 25$	0.35	0.92	0.94
$n_{noise} = 50$	0.35	0.90	0.95
(Base) $n_{noise} = 100$	0.33	0.93	0.96
(Base) XGBoost	0.33	0.93	0.96
LightGBM	0.33	0.92	0.97
CatBoost	0.64	0.56	0.81
Random Forests	0.94	0.34	0.78
(Base) $y_{cond} = False$	0.33	0.93	0.96
$y_{cond} = True$	0.33	0.93	0.97

Results are presented in Table 3. We find that the best overall methods are MICE-Forest and MissForest; both methods are nearly equally good, with the difference being that MissForest has low diversity and is thus best for single imputation. These results highlight the power of tree-based methods on tabular data.

For the remaining methods, it is hard to rank them globally since they vary in performance across the different metrics. With raw scores, our method generally outperforms them (See Table 9 and bar plots in §C.2.2). However, with rank scores, our method can rank above or below GAIN, OT, and KNN, depending on the metrics. However, we note that our method is much more diverse than other methods, making it appealing for multiple imputation strategies.

4.3 Generation with incomplete tabular data

For generation of incomplete data, we follow the same setup as the generation with complete data (see Section 4.1) but randomly remove 20% of the values in the input X (MCAR). Our method can work on incomplete data directly thanks to XGBoost, so we do not impute the data. For other methods, we first impute the input data X with MissForest prior to training (we could also follow this strategy for our Forest Flow and Diffusion methods).

The averaged rank of each method is presented in Table 4, their raw scores in Table 10 and the bar plots in C.2.3. Forest-Flow and Forest-VP perform similarly and are generally the highest-performing methods except on training coverage, R-squared, and coverage rate metrics, where TabDDPM is slightly better. These results show that our method performs similarly or better than their deep-learning counterpart without needing GPUs and by training directly on missing data instead of relying on missing data imputation.

4.4 Ablation of Forest-VP and Forest-Flow

We run an ablation on the Iris dataset when varying the hyperparameters of the algorithm and the choice of tree-based approximator (with default hyperparameters and L1/L2 penalizations disabled). The results can be found for Forest-Flow in Table 5 and for Forest-Diffusion in Table 6. For both flow and diffusion, the best performance is obtained at $n_t = 50$, and the larger n_{noise} is. XGBoost and LightGBM are superior to other tree-based methods. However, LightGBM freezes when running multiple models in parallel, and

training not-in-parallel is much slower than training XGBoost in parallel; thus, given its significant speed advantage, we use XGBoost as our main GBT method. We observe that conditioning on the outcome label improves performance for some metrics, but the difference is not significative.

5 CONCLUSION

We presented the first approach to train diffusion and flow-based models using XGBoost (and other Gradient-Boosted Tree methods) instead of neural networks. Our method generates highly realistic synthetic tabular data even when the training data contains missing values and they also generate diverse imputations. Our method performs better or on-par to deep-learning methods without requiring GPUs. Our work shows that tree-based methods are currently the best algorithms to deal with tabular data on data generation tasks. Our Python (Van Rossum and Drake, 2009) and R (R Core Team, 2021) code is available at `https://github.com/SamsungSAILMontreal/ForestDiffusion`.

While powerful, our method has limitations. Although we obtain state-of-the-art results on the generation task, MICE-Forest and MissForest remain better for imputation. Furthermore, because of not using mini-batch training, the memory demand of GBTs can potentially be higher than deep-learning methods for large datasets. This limitation is further exacerbated by the fact that we duplicate the rows of the dataset n_{noise} times, thus making the dataset much bigger.

As future work, we could potentially use multinomial diffusion (Hoogeboom et al., 2021) to improve performance. We could also consider using new diffusion processes (Karras et al., 2022) or classifier-free guidance (Ho and Salimans, 2022). It would be helpful to find a way to train Tree models with mini-batches to remove the need to train on a duplicated dataset (similarly to the novel data iterator from XGBoost 2.0). As future applications, it would be interesting to apply this technique to data-augmentation, class imbalance, and domain translation tasks.

To better understand the data generation process, one could extract feature importance across all XGBoost models (an example is shown in Figure 4 for Forest-Flow). This could also be used for variable selection.

6 BROADER IMPACTS

Our method generates new observations that closely match the data's true observations. Although the synthetic samples are highly realistic, they are still generated. One should be careful about drawing inferences on fake participants, considering their potential real-world impact. We encourage practitioners to always compare fake data inference to real data inference.

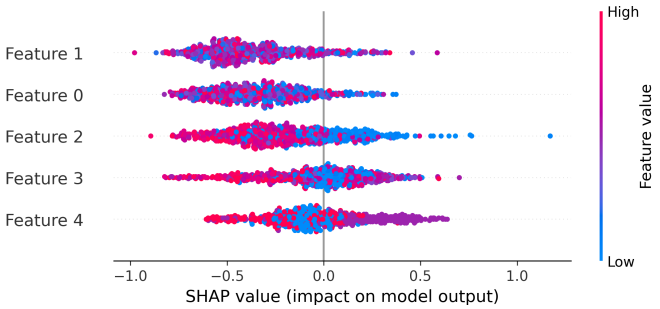


Figure 4: Average feature importance (SHAP value) across all XGBoost models trained with Forest-Flow on Iris dataset. The features (from 0 to 4) are, respectively, sepal length, sepal width, petal length, petal width, and species. Instead of average, one could also get a different plot per noise-level or variable predicted.

Missing data imputation is ultimately an *elaborate guess*. Imputing data is sensible when one has few missing values per participant or when non-missing values are highly correlated to missing values. Otherwise, imputation methods effectively fabricate most of the information about an observation (or person), and the practitioner may draw conclusions based on this information. We encourage practitioners to 1) use multiple imputations to average over different plausible imputations when making inferences and 2) not use imputations when they have too much missing data (get more participants or find ways to retrieve the missing data).

Perhaps one of the biggest advantages of our method is its conservative nature with respect to resources. Typical diffusion models and generative AI algorithms are extremely resource intensive, taking special dedicated hardware as well as distributed training. Meanwhile, our method can be run on a regular laptop (with 4-12 CPUs). This is in extreme contrast to most deep neural networks, which require expensive GPUs.

Acknowledgments

This research was enabled in part by compute resources provided by Mila (`mila.quebec`). KF is supported by NSERC Discovery grant (RGPIN-2019-06512), a CIFAR AI Chair and a Samsung grant. TK is supported by the Ethereum foundation and Lineage logistics foundation grant. We thank Simon Lacoste-Julien, Quentin Bertrand and Emy Gervais for their valuable input.

References

Congressional Voting Records. UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C5C01P.

Stefan Aeberhard and M. Forina. Wine. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5PC7J.

David Aha. Tic-Tac-Toe Endgame. UCI Machine Learning Repository, 1991. DOI: https://doi.org/10.24432/C5688J.

Michael Samuel Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=li7qeBbCR1t`.

Bhattarai Ashim. missingpy. `https://github.com/epsilon-machine/missingpy`, 2013.

Derrick A Bennett. How can i deal with missing data in my study? *Australian and New Zealand journal of public health*, 25(5):464–469, 2001.

Rajen Bhatt. Planning Relax. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5T023.

Marko Bohanec. Car Evaluation. UCI Machine Learning Repository, 1997. DOI: https://doi.org/10.24432/C5JP48.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022a.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*, 2022b.

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

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Thomas Brooks, D. Pope, and Michael Marcolini. Airfoil Self-Noise. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5VW2C.

Samuel F Buck. A method of estimation of missing values in multivariate data suitable for use with an electronic computer. *Journal of the Royal Statistical Society: Series B (Methodological)*, 22(2):302–306, 1960.

Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7608–7617, 2019.

Magorzata Charytanowicz, Jerzy Niewczas, Piotr Kulczycki, Piotr Kowalski, and Szymon Lukasik. seeds. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5H30K.

Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*, 2018.

Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Wine Quality. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C56S3T.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International conference on machine learning*, pages 1174–1183. PMLR, 2018.

David Deterding, Mahesan Niranjan, and Tony Robinson. Connectionist Bench (Vowel Recognition - Deterding Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C58P4S.

Daniel Dias, Sarajane Peres, and Helton Bscaro. Libras Movement. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5GC82.

Allan Donner. The relative effectiveness of procedures commonly used in multiple regression analysis for dealing with missing values. *The American Statistician*, 36(4):378–381, 1982.

Dheeru Dua and Casey Graff. Uci machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Kilian Fatras, Younes Zine, Rémi Flamary, Remi Gribonval, and Nicolas Courty. Learning with minibatch wasserstein : asymptotic and gradient properties. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International*

Conference on Artificial Intelligence and Statistics, volume 108 of *Proceedings of Machine Learning Research*, pages 2131–2141, Online, 26–28 Aug 2020. PMLR. URL http://proceedings.mlr.press/v108/fatras20a.html.

Kilian Fatras, Younes Zine, Szymon Majewski, Rémi Flamary, Rémi Gribonval, and Nicolas Courty. Minibatch optimal transport distances; analysis and applications, 2021.

William Feller. On the theory of stochastic processes, with particular reference to applications. 1949. URL https://api.semanticscholar.org/CorpusID:121027442.

R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.

Piotr Florek and Adam Zagdański. Benchmarking state-of-the-art gradient boosting algorithms for classification. *arXiv preprint arXiv:2305.17094*, 2023.

Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. In *International Conference on Machine Learning*, pages 3233–3246. PMLR, 2020.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Aude Genevay, Gabriel Peyre, and Marco Cuturi. Learning generative models with sinkhorn divergences. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1608–1617. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/genevay18a.html.

B. German. Glass Identification. UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C5WW2P.

J. Gerritsma, R. Onnink, and A. Versluis. Yacht Hydrodynamics. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5XG7R.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

John C Gower. A general coefficient of similarity and some of its properties. *Biometrics*, pages 857–871, 1971.

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

William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos. *Advances in Neural Information Processing Systems*, 35:27953–27965, 2022.

Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Neural Information Processing Systems (NeurIPS)*, 2020.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Harry Joe. *Dependence modeling with copulas*. CRC press, 2014.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. *arXiv preprint arXiv:2210.04018*, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Murat Koklu and Ilker Ali Ozkan. Multiclass classification of dry beans using computer vision and machine learning techniques. *Computers and Electronics in Agriculture*, 174:105507, 2020.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

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.

Ranjit Lall and Thomas Robinson. The midas touch: accurate and scalable missing-data imputation with deep learning. *Political Analysis*, 30(2):179–196, 2022.

Peng Li, Elizabeth A Stuart, and David B Allison. Multiple imputation: a flexible tool for handling missing data. *Jama*, 314(18):1966–1967, 2015.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, October 2022.

Max Little. Parkinsons. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C59C74.

Roderick JA Little. Missing-data adjustments in large surveys. *Journal of Business & Economic Statistics*, 6(3):287–296, 1988.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 1987.

Roderick JA Little and Donald B Rubin. The analysis of social science data with missing values. *Sociological methods & research*, 18(2-3):292–326, 1989.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

DD Lucas, R Klein, J Tannahill, D Ivanova, S Brandon, D Domyancic, and Y Zhang. Climate Model Simulation Crashes. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5HG71.

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*, pages 11461–11471, 2022.

Baoshan Ma, Fanyu Meng, Ge Yan, Haowen Yan, Bingjie Chai, and Fengju Song. Diagnostic classification of cancers using extreme gradient boosting algorithm and multi-omics data. *Computers in biology and medicine*, 121:103761, 2020.

Marcos Roberto Machado, Salma Karray, and Ivaldo Tributino de Sousa. Lightgbm: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry. In *2019 14th International Conference on Computer Science & Education (ICCSE)*, pages 1111–1116. IEEE, 2019.

Kamel Mansouri, Tine Ringsted, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. QSAR biodegradation. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5H60M.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16:100258, 2022. ISSN 2590-0056. doi: https://doi.org/10.1016/j.array.2022.100258. URL `https://www.sciencedirect.com/science/article/pii/S2590005622000911`.

Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing data imputation using optimal transport. In *International Conference on Machine Learning*, pages 7130–7140. PMLR, 2020.

Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International Conference on Machine Learning*, pages 7176–7185. PMLR, 2020.

Kenta Nakai. Ecoli. UCI Machine Learning Repository, 1996a. DOI: https://doi.org/10.24432/C5388M.

Kenta Nakai. Yeast. UCI Machine Learning Repository, 1996b. DOI: https://doi.org/10.24432/C5KG68.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

Yidong Ouyang, Liyan Xie, Chongxuan Li, and Guang Cheng. Missdiff: Training diffusion models on tabular data with missing values. *arXiv preprint arXiv:2307.00467*, 2023.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3): 291–297, 1997.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016. doi: 10.1109/DSAA.2016.49.

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

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Halsey Lawrence Royden and Patrick Fitzpatrick. *Real analysis*, volume 2. Macmillan New York, 1968.

Terry Sejnowski and Paul R Gorman. Connectionist Bench (Sonar, Mines vs. Rocks). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T01Q.

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

Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Ionosphere. UCI Machine Learning Repository, 1989. DOI: https://doi.org/10.24432/C5W01B.

M Sklar. Fonctions de répartition à n dimensions et leurs marges. In *Annales de l'ISUP*, volume 8, pages 229–231, 1959.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Neural Information Processing Systems (NeurIPS)*, 2019.

Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. *Neural Information Processing Systems (NeurIPS)*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *International Conference on Learning Representations (ICLR)*, 2021.

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

Daniel B Suits. Use of dummy variables in regression equations. *Journal of the American Statistical Association*, 52(280):548–551, 1957.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

Alexander Tong, Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio. Simulation-free schrödinger bridges via score and flow matching, 2023a.

Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian Fatras, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint 2302.00482*, 2023b.

Samir Touzani, Jessica Granderson, and Samuel Fernandes. Gradient boosting machine for modeling the

energy consumption of commercial buildings. *Energy and Buildings*, 158:1533–1543, 2018.

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

Stef Van Buuren. *Flexible imputation of missing data*. CRC press, 2018.

Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011.

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.

Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in Neural Information Processing Systems*, 35:23371–23385, 2022.

Zhenhua Wang, Olanrewaju Akande, Jason Poulos, and Fan Li. Are deep learning models superior for missing data imputation in surveys? evidence from an empirical comparison. *Survey Methodology*, 48 (2):375–399, 2022.

Jeremy Watt, Reza Borhani, and Aggelos K Katsaggelos. *Machine learning refined: Foundations, algorithms, and applications*. Cambridge University Press, 2020.

Samuel Von Wilson, Bogdan Cebere, James Myatt, and Samuel Wilson. miceforest, 2023. URL `https://github.com/AnotherSamWilson/miceforest/`.

William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

I-Cheng Yeh. Concrete Compressive Strength. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C5PK67.

I-Cheng Yeh. Blood Transfusion Service Center. UCI Machine Learning Repository, 2008. DOI: https://doi.org/10.24432/C5GS39.

I-Cheng Yeh. Concrete Slump Test. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5FG7D.

Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pages 5689–5698. PMLR, 2018.

Taeyoung Yun, Haewon Jung, and Jiwoo Son. Imputation as inpainting: Diffusion models for spatiotemporal data imputation. 2023.

Chongsheng Zhang, Changchang Liu, Xiangliang Zhang, and George Almpanidis. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, 2017.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan: Effective table data synthesizing. In *Asian Conference on Machine Learning*, pages 97–112. PMLR, 2021.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y Chen. Ctab-gan+: Enhancing tabular data synthesis. *arXiv preprint arXiv:2204.00401*, 2022.

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

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary material: Generating and Imputing Tabular Data via Diffusion and Flow-based Gradient-Boosted Trees

The supplementary material is structured as follows:

- Appendix A describes the full algorithms of our methods Forest-VP and Forest-Flow.
- Appendix B describes the datasets, the metrics and the competitors that we used in our experiments.
- Appendix C reports all averaged raw scores over the 27 datasets as well as the bar plots for each dataset, metric and method.

A ALGORITHMS

A.1 REPAINT Imputation Algorithm

Algorithm 4: Forest-Diffusion Imputation with REPAINT

Input: XGBoost models $\{f(t)\}_{t\in t_{levels}}$, dataset X of size $[n_{obs}, d]$, d variables, $n_{noise} = 100$, $n_t = 50$ noise levels, $r = 10$ repaints, jump size $j = 5$.

$M \leftarrow$ Mask indicating which values are non-missing in X (1: non-missing, 0: missing)

$X(1) \leftarrow$ Dataset of $z \sim \mathcal{N}(0, 1)$ with size $[n_{obs}, d]$

$n_{repaint} \leftarrow 1$

$t \leftarrow 1$

while $t > 0$ **do**

  $X(t - \frac{1}{n_t}) \leftarrow$ Empty Dataset with size $[n_{obs}, d]$

  `// Reverse process for missing values`

  $X(t - \frac{1}{n_t})[1 - M] \leftarrow \text{Reverse}(X(t)[1 - M] \mid f(t), t, n_t)$

  `// Set non-missing values to truth`

  $X(t - \frac{1}{n_t})[M] \leftarrow \text{Forward}(Z[M], X[M], 0, t \mid n_t)$

  if $n_{repaint} < r$ *and* $t + 1 \mod j = 0$ **then**

    `// go back a few steps to repaint`

    $n_{repaint} \leftarrow n_{repaint} + 1$

    $Z \leftarrow$ Dataset of $z \sim \mathcal{N}(0, 1)$ with size $[n_{obs}, d]$

    $X(t + \frac{j}{n_t})[M] \leftarrow \text{Forward}(Z[M], X(t)[M], t, t + \frac{j}{n_t}, \mid n_t)$

    $t \leftarrow t + \frac{j}{n_t}$

  end

  if $n_{repaint} = r$ *and* $n_t t + 1 \mod j = 0$ **then**

    $n_{repaint} \leftarrow 0$ `// finished repainting`

  end

  $t \leftarrow t - \frac{1}{n_t}$

end

return $X(0)$

A.2 Forward and Reverse step Algorithms

Algorithm 5: Forward $(0 \to t)$

Input: Noise data Z, dataset X, time t, number of noise levels n_t, boolean $flow$ (if True, use conditional flow matching; if False, use VP diffusion), $\beta = [0.1, 20]$.

if $flow$ **then**
- $X(t) \leftarrow (1-t)Z + tX$
- $Y(t) \leftarrow X - Z$

end
else
- $C \leftarrow -\frac{1}{4}t^2(\beta[1] - \beta[0]) - 0.5t\beta[0]$
- $X(t) \leftarrow \exp CX + \sqrt{1 - \exp 2C}Z$
- $Y(t) \leftarrow Z$

end
return $X(t), Y(t)$

Algorithm 6: Forward $(t \to t_{next})$

Input: Noise data Z, dataset X, next time t_{next} (where $t_{next} > t$) , current time t, number of noise levels n_t, boolean $flow$ (if True, use conditional flow matching; if False, use VP diffusion), $\beta = [0.1, 20]$.

$h \leftarrow t_{next} - t$
if $flow$ **then**
- $X(t_{next}) \leftarrow Z + h(X - Z)$

end
else
- $\beta_t = \beta[0] + t(\beta[1] - \beta[0])$
- $X(t_{next}) \leftarrow X(t) - h\frac{1}{2}\beta_t X(t) + \sqrt{\beta_t}Z$

end
return $X(t_{next})$

Algorithm 7: Reverse

Input: dataset $X(t)$, time t, XGBoost model $f(t)$, number of noise levels n_t, boolean $flow$ (if True, use conditional flow matching; if False, use VP diffusion), $\beta = [0.1, 20]$.

$h \leftarrow \frac{1}{n_t}$
if $flow$ **then**
- $X(t - \frac{1}{n_t}) \leftarrow X(t) + hf(t)$

end
else
- $\beta_t = \beta[0] + t(\beta[1] - \beta[0])$
- $C \leftarrow -\frac{1}{4}t^2(\beta[1] - \beta[0]) - 0.5t\beta[0]$
- $score \leftarrow -\frac{f(t)}{\sqrt{1 - \exp 2C}}$
- $\mu = -\frac{1}{2}\beta_t X(t) - \beta_t score$
- $X(t - \frac{1}{n_t}) \leftarrow X(t) - h\mu + \beta_t\sqrt{h}Z$

end
return $X(t - \frac{1}{n_t})$

A.3 Flow matching

Flow matching has been introduced by several works under different names Lipman et al. (2022); Albergo and Vanden-Eijnden (2023); Liu et al. (2022). As explained in the main text, Flow Matching's objective is to regress a conditional vector field built from conditional probability paths. In this section, we provide more details about the conditional probability paths and conditional vector fields that were used respectively in Lipman et al. (2022) and Tong et al. (2023b).

The natural choice of the conditional probability path $p_t(x|z)$ is a Gaussian conditioned on a latent variable $z \sim q(z)$ with variance σ_t leading to $p_t(x) = \int \mathcal{N}(\nu_t(z), \sigma_t)q(z)dz$. In particular, we want that p_1 approximates the distribution q. When we use a Gaussian for the conditional probability path, it is possible to compute the conditional vector field in closed-form thanks to the next Theorem:

Theorem 1 (Theorem 3 of Lipman et al. (2022))**.** *The unique vector field whose integration map satisfies* $\rho_t(x) = \nu_t + \sigma_t x$ *has the form*

$$\mu_t(s) = \frac{\sigma_t'}{\sigma_t}(s - \nu_t) + \nu_t', \tag{6}$$

Therefore to develop a conditional flow matching variant, we have to choose four different quantities: the conditioning z, the Gaussian mean ν_t, the Gaussian standard deviation σ_t and the latent distribution q.

Flow Matching. We first consider the condition z to be a single training sample $z = x_1$, the Gaussian mean is $\nu_t(x_1) = tx_1$ and its standard deviation $\sigma_t = (t\sigma - t + 1)^2$ where $\sigma > 0$. Regarding q, we set it to $q(z) = q(x_1)$ as the uniform distribution over the training dataset. Now we define the conditional probability path and the conditional vector field as:

$$p_t(x|z) = \mathcal{N}(tx_1, (t\sigma - t + 1)^2), \tag{7}$$

$$\mu_t(x|z) = \frac{x_1 - (1-\sigma)x}{1 - (1-\sigma)t}, \tag{8}$$

which is a probability path from the standard normal distribution ($p_0(x|z) = \mathcal{N}(x; 0, 1)$) to a Gaussian distribution centered at x_1 with standard deviation σ ($p_1(x|z) = \mathcal{N}(x; x_1, \sigma^2)$) in order to ensure that $p_1 \approx q$.

Independent-Conditional Flow Matching. Another recent Flow Matching variant is *Independent-Conditional Flow Matching*. In this variant, the condition z is a tuple of a source and a target sample $z = (x_0, x_1)$, the mean of Gaussian conditional probability path is set to $\nu_t(x_0, x_1) = tx_1 + (1-t)x_0$, the standard deviation σ_t to a constant independent of t and the latent distribution to the independent coupling $q(x_0, x_1) = q(x_0)q(x_1)$.

$$p_t(x|x_0, x_1) = \mathcal{N}(x \mid tx_1 + (1-t)x_0, \sigma^2), \tag{9}$$

$$p_t(x) = \int \mathcal{N}(x \mid tx_1 + (1-t)x_0, \sigma^2)\pi(x_0, x_1)dx_0dx_1, \tag{10}$$

$$\mu_t(x|x_0, x_1) = x_1 - x_0. \tag{11}$$

We note that we also recover $p_1(x|z) = \mathcal{N}(x; x_1, \sigma^2)$ as the above Flow Matching variant. The I-CFM loss is equal to $\mathcal{L}_{\text{Forest-Flow}} = \|v_\theta(t, x) - \mu_t(x|z)\| = \|v_\theta(t, x) - (x_1 - x_0)\|$. The advantage of this variant is that it is simple to implement and available in open-source in the TorchCFM package[3] Tong et al. (2023b). Note that in our experiments, we have set σ to 0.

A.4 Why Flow methods cannot be used for imputation or inpainting

We considered doing imputation for flow models through regular inpainting or REPAINT. However, we found that it could not work due to the deterministic nature of the method, as explained below.

A flow is such that a single Gaussian sample $x(1) = z$ (of dimension d) follows a deterministic trajectory leading to a single data sample $x(0)$. For imputation, we need the final $x(0)$ to have the same non-missing values as the true sample containing missing data. However, since the trajectory from $x(1)$ to $x(0)$ is deterministic, we have no control over which data sample we will end up with at $t = 0$. Trying to force $x(t)$ into the direction that we want (toward the true non-missing values) does not work because we diverge from the actual deterministic trajectory. Therefore, for imputation, we had to rely exclusively on diffusion methods.

[3] `https://github.com/atong01/conditional-flow-matching`

B DATASETS, METRICS AND METHODS

B.1 Datasets

We list all the datasets in Table 7. We mostly use the same datasets as Muzellec et al. (2020) (with the exception of *bean*), and the datasets come either from the UCI Machine Learning Repository (Dua and Graff, 2017) or scikit-learn (Pedregosa et al., 2011). All UCI datasets are licensed under the Creative Commons Attribution 4.0 International license (CC BY 4.0). For scikit-learn, the Iris dataset is licensed with the BSD 3-Clause License, and the California housing is freely provided by the authors with no license. Irrespective of the source, all datasets are openly shared with no restriction on usage.

Table 7: Datasets

Dataset	Citation	n	d	Outcome
airfoil self noise	(Brooks et al., 2014)	1503	5	continuous
bean	(Koklu and Ozkan, 2020)	13611	16	categorical
blood transfusion	(Yeh, 2008)	748	4	binary
breast cancer diagnostic	(Wolberg et al., 1995)	569	30	binary
california housing	(Pace and Barry, 1997)	20640	8	continuous
climate model crashes	(Lucas et al., 2013)	540	18	binary
concrete compression	(Yeh, 2007)	1030	7	continuous
concrete slump	(Yeh, 2009)	103	7	continuous
connectionist bench sonar	(Sejnowski and Gorman)	208	60	binary
connectionist bench vowel	(Deterding et al.)	990	10	binary
ecoli	(Nakai, 1996a)	336	7	categorical
glass	(German, 1987)	214	9	categorical
ionosphere	(Sigillito et al., 1989)	351	34	binary
iris	(Fisher, 1988)	150	4	categorical
libras	(Dias et al., 2009)	360	90	categorical
parkinsons	(Little, 2008)	195	23	binary
planning relax	(Bhatt, 2012)	182	12	binary
qsar biodegradation	(Mansouri et al., 2013)	1055	41	binary
seeds	(Charytanowicz et al., 2012)	210	7	categorical
wine	(Aeberhard and Forina, 1991)	178	13	categorical
wine quality red	(Cortez et al., 2009)	1599	10	integer
wine quality white	(Cortez et al., 2009)	4898	11	integer
yacht hydrodynamics	(Gerritsma et al., 2013)	308	6	continuous
yeast	(Nakai, 1996b)	1484	8	categorical
tic-tac-toe	(Aha, 1991)	958	9	binary
congressional voting	(mis, 1987)	435	16	binary
car evaluation	(Bohanec, 1997)	1728	6	categorical

B.2 Metrics

We describe below the choices of metrics used in this paper.

B.2.1 Gower distance

Since we have mixed-type data (continuous and categorical variables) at varying scales (one variable can be thousands and one in decimals), the Euclidean distance is not an adequate distance measure. We take inspiration from the k-Nearest Neighbors (KNN) literature by instead relying on the Gower distance (Gower, 1971), a popular metric to uniformize distances between continuous and categorical variables. The Gower distance is the sum of the per-variable costs, and these costs are defined to always be between 0 and 1 for both continuous and categorical variables. We use this distance for every evaluation metric that relies on a distance metric (i.e.,

Wasserstein distance, coverage, and MAE). The Gower distance can be implemented by taking the L1 distance over min-max normalized continuous variables or one-hot categorical variables divided by two.

B.2.2 Imputation

Wasserstein Distance For imputation, to assess how close the real and imputed data distributions are, we use the Wasserstein distance. We report the Wasserstein distance to either the training data or testing data. Since calculating the Wasserstein distance scales quadratically with sample size, to the reduce time taken, we only calculate the distance for datasets with less than 5000 training samples. This approach is also chosen by Muzellec et al. (2020).

Mean Absolute Deviation around the median/mode (MAD) To assess the diversity of the imputations, we look at the Mean Absolute Deviation (MAD) around the median (for continuous variables) or mode (for categorical variables) of imputed values across $k = 5$ imputations. MAD measures the variability using the L1 distance to the median/mode, which is more sensible than using the variance given our reliance on the Gower Distance, which is L1-based. This measure says nothing about quality, but given equal levels of quality, one should prefer a method that is more diverse in its imputations to avoid creating false precision and ensure valid uncertainty on parameter estimates (Li et al., 2015).

Minimum and Average MAE A common approach to assess the distance between the imputed values and the ground-truth values is the root mean squared error (RMSE) or mean absolute error (MAE) (Stekhoven and Bühlmann, 2012). However, the usefulness of these metrics has been severely debated (Van Buuren, 2018; Wang et al., 2022). An imputation method that accounts for uncertainty by producing multiple different imputations will inherently have a higher RMSE/MAE than one centered across the mean or median of possible imputations. We see this in Tashiro et al. (2021), where instead of directly taking the multiple imputations they generate, the authors only report the median imputation of their 100 different imputations to be as close as possible to the ground truth and thus obtain a great RMSE/MAE; however, in doing so, they destroy all the sampling diversity produced by their method. Furthermore, this approach is only usable when the number of variables is 1, given that no exact concept of multivariate median exists.

Since taking the median approach of Tashiro et al. (2021) cannot work in the multivariate setting, we adjust the metric through the standard approach used in the video prediction literature (Denton and Fergus, 2018; Castrejon et al., 2019; Franceschi et al., 2020; Voleti et al., 2022), which is to produce different samples ($k = 5$ different imputations for a given missing observation in our case) and take the distance between the ground truth and the sample closest to this ground truth. The idea is that different imputations may still be plausible and valid even though they differ from the ground truth. Hence, this approach only tries to ensure that the ground truth is at least close to one of the possible imputations (rather than to all the possible imputations).

While the average MAE is unfair against stochastic methods, taking the minimum may be considered unjust against deterministic methods (because it gives more chance for the stochastic method to be closer by increasing the number of imputations k). Thus, we report both the minimum and the average MAE. We use the MAE instead of the RMSE because it matches the Gower Distance, which relies on the L1 distance.

Efficiency To assess the practical use for machine learning purposes, we use the efficiency/utility (Xu et al., 2019); this measure is defined as the average F1-score for classification problems and R^2 for regression problems for models trained on imputed data and evaluated on the test set.

We average the efficiency over four useful non-deep models: linear/logistic regression, AdaBoost, Random Forests, and XGBoost. The choice of models used is a matter of taste. Kotelnikov et al. (2023) include CatBoost instead of XGBoost. Xu et al. (2019) do not include any Gradient-Boosted Trees (GBTs), but they include both Decision Trees (DTs) and Multilayer perceptrons (MLPs). We prefer not to use DTs because a single tree has limited capacity, and almost no one uses them in practice (over GBTs or Random Forests). We also prefer not to use MLPs since data scientists and statisticians rarely use them for tabular data given their massive complexity, computational expense (GPUs), and generally inferior performance (Shwartz-Ziv and Armon, 2022).

Statistical measures To assess the ability to obtain statistically valid inferences from incomplete data, we rely on classic statistical measures which train a linear regression model and compare the parameter estimates

obtained by the imputed data to the ones obtained through the ground-truth data with no missing values (Van Buuren, 2018). The metrics we use are: the percent bias ($|\mathbb{E}\left[\frac{\hat{\beta}-\beta}{\beta}\right]|$), and the coverage rate (i.e., the proportion of confidence intervals containing the true value).

We report the average of those measures across all regression parameters. These metrics focus on ensuring that the imputed data do not lead to wrong incorrect assessments about the importance of each variable during inference. For example, if one seeks to know if a certain gene influence a certain outcome, one needs to make sure the imputed data does not incorrectly give a significant effect for this gene when there isn't one (or the converse).

B.2.3 Generation

Wasserstein Distance For generation, to assess how close the real and fake data distributions are, we use the Wasserstein distance. We report the Wasserstein distance to either the training data or testing data. Since calculating the Wasserstein distance scales quadratically with sample size, to the reduce time taken, we only calculate the distance for datasets with less than 5000 training samples. This approach is also taken by Muzellec et al. (2020).

Coverage To assess the diversity of generated samples, we use the coverage (Naeem et al., 2020), a measure of the ratio of real observations that have at least one fake observation within a sphere of radius r, where r is the distance between the sample and its k-th nearest neighbor. The number of nearest neighbors k is the smallest value such that we obtain at least 95% coverage on the true data. We report the coverage for both training and testing data separately.

Efficiency We again use the average efficiency when training on synthetic data to assess the practical use for machine learning purposes.

Discriminator We follow the same design as the Efficiency F1 score metric but use a classifier to predict if a sample is real or fake. We test the classifier on fake data and report the F1 score. A high F1 score means the classifier recognizes the fake data as fake; thus, lower values correspond to better generation quality. Note that we train with the training dataset and fake data of the same size but test on another random sample of fake data to counter overfitting.

Statistical measures We use the same statistical measures while using datasets filled with fake samples instead of imputed datasets.

B.3 Details of each methods

B.3.1 Computational resources used

We trained the tree-based models on a cluster of 10-20 CPUs with 64-256Gb of RAM.

We trained the other models on a cluster with 8 CPUs, 1 GPU, and 48-128Gb of RAM.

B.3.2 Generation

Oracle This is the actual training data.

ForestFlow and ForestVP We use $n_t = 50$, $n_{noise} = 100$, label conditioning, and dummy encoding of the categorical variables.

GaussianCopula GaussianCopula transforms the data into a copula, a distribution on the unit cube (Sklar, 1959). Then, it can generate new data by sampling from the unit cube and reversing the copula. We use the Gaussian Copula implementation from the Synthetic data vault (Patki et al., 2016) with the default hyperparameters.

TVAE and CTGAN Tabular VAE (TVAE) is a Variational AutoEncoder (VAE) (Kingma and Welling, 2013) for mixed-type tabular data generation. Conditional Tabular GAN (CTGAN) is a Generative Adversarial Network (GAN) (Goodfellow et al., 2014) for mixed-type tabular data generation. Both TVAE and CTGAN were introduced by Xu et al. (2019). We use the implementation from the Synthetic data vault (Patki et al., 2016) with the default hyperparameters.

CTAB-GAN+ CTAB-GAN (Zhao et al., 2021) is a Generative Adversarial Network (GAN) (Goodfellow et al., 2014) for mixed-type tabular data generation. CTAB-GAN+ is an improvement of the method (Zhao et al., 2022). We use the official implementation with the default hyperparameters.

STaSy Score-based Tabular data Synthesis (STaSy) (Kim et al., 2022) is a method using score-based generative models (Song et al., 2021) for mixed-type tabular data generation. We use the official implementation. We initially obtained nonsensical outputs (lots of variables at the min or max value and/or generating only one class) on small data generation (such as with the Iris dataset) with STaSy default hyperparameters . Through correspondence with the authors, we got STaSy to work better through small changes in the hyperparameters (using Naive STaSy, 10000 epochs, and reducing the hidden dimensions to $(64, 128, 256, 128, 64)$). We use the default hyperparameters with those changes.

Tab-DDPM Tab-DDPM (Kotelnikov et al., 2023) is a method using denoising diffusion probabilistic models (Ho et al., 2020) for mixed-type tabular data generation. We use the official implementation with the default hyperparameters from the California dataset.

B.3.3 Imputation

Oracle This is the actual training data before adding missing values.

Forest-Flow and Forest-VP We use $n_t = 50$, $n_{noise} = 100$ (except for the *bean*, where we use $n_{noise} = 50$ to reduce memory requirements), and dummy encoding of the categorical variables.

KNN We use the k-nearest neighbors (KNN) (Cover and Hart, 1967) based imputation method by Troyanskaya et al. (2001). We use the scikit-learn (Pedregosa et al., 2011) implementation with $k = 1$ and default hyperparameters. There is no L1-based KNN imputer to our knowledge, so we cannot work with the Gower Distance. Thus, we stick to the regular KNN imputer, which depends on a special Euclidean distance made for incomplete data that penalizes distance based on the number of missing values. Since the distance is L2-based, we standardize (z-score) all the variables prior to being imputed and unstandardize them after imputation.

ICE Imputation by Chained Equations (ICE) does iterative imputations through conditional expectation. We use the IterativeImputer function from scikit-learn (Pedregosa et al., 2011), which is based on Van Buuren and Groothuis-Oudshoorn (2011). We use 10 iterations and the default hyperparameters. We clip the imputations to be between the minimum and maximum of observed values.

MICE-Forest MICE-Forest (also called miceRanger) (Wilson et al., 2023) is an imputation method using Multiple Imputations by Chained Equations (MICE) (Van Buuren and Groothuis-Oudshoorn, 2011) with predictive mean matching (Little, 1988) and LightGBM (Ke et al., 2017), a popular type of Gradient-Boosted Tree (GBT) method. We use the official Python library with the default hyperparameters.

MissForest MissForest (Stekhoven and Bühlmann, 2012) is an iterative algorithm using Random Forests (Breiman, 2001) to impute missing data. We use the implementation from the *missingpy* Python library (Ashim, 2013) with the default hyperparameters.

Softimpute Softimpute (Hastie et al., 2015) is an iterative soft-threshold Singular Value Decomposition (SVD) method to impute missing data. We use the implementation from Muzellec et al. (2020) with the default hyperparameters. We clip the imputations to be between the minimum and maximum of observed values.

Sinkhorn Minibatch Sinkhorn divergence (Muzellec et al., 2020) is an Optimal Transport (OT) (Villani et al., 2009) method based on the minibatch Sinkhorn divergence (Cuturi, 2013; Genevay et al., 2018) to impute missing

data. The intuition is that two minibatches from the same distribution should have relatively similar statistics. Therefore, they leverage the minibatch optimal transport Fatras et al. (2020, 2021) loss to imput missing data. We use the official implementation with the default hyperparameters. We clip the imputations to be between the minimum and maximum of observed values.

GAIN Generative Adversarial Imputation Nets (GAIN) (Yoon et al., 2018) is a GAN (Goodfellow et al., 2014) based method to impute missing data. We use the official implementation with the default hyperparameters. The code does rounding in a problematic way; it assumes that variables with less than 20 unique values are categorical, and those variables are rounded to the nearest integer. We found this to cause problems, so we instead provide the names of the categorical variables to the function so that only those variables are rounded.

C AVERAGED RESULSTS AND BAR PLOTS

In this section, we provide the averaged raw score for each method on all datasets as well as the different bar plot for each metric and each method on each dataset.

C.1 Tables - raw score

In this section, we provide the averaged raw score for each method on all datasets for the generation of tabular data 8, the tabular data imputation 9 and the tabular data generation with missing values 10.

Table 8: Tabular data generation with complete data (27 datasets, 3 experiments per dataset); raw score - mean (standard-error)

	$W_{train}\downarrow$	$W_{test}\downarrow$	$cov_{train}\uparrow$	$cov_{test}\uparrow$	$R^2_{fake}\uparrow$	$F1_{fake}\uparrow$	$F1_{disc}\downarrow$	$P_{bias}\downarrow$	$cov_{rate}\uparrow$
GaussianCopula	2.74 (0.56)	2.99 (0.61)	0.18 (0.04)	0.37 (0.06)	0.20 (0.14)	0.46 (0.06)	0.62 (0.05)	2.27 (0.77)	0.23 (0.12)
TVAE	2.12 (0.58)	2.35 (0.63)	0.33 (0.04)	0.63 (0.04)	-0.47 (0.61)	0.52 (0.08)	0.44 (0.01)	4.15 (1.97)	0.26 (0.09)
CTGAN	3.58 (0.99)	3.74 (1.01)	0.12 (0.03)	0.28 (0.04)	-0.43 (0.08)	0.35 (0.04)	0.48 (0.01)	2.48 (1.30)	0.20 (0.08)
CTAB-GAN+	2.71 (0.81)	2.89 (0.83)	0.22 (0.04)	0.44 (0.05)	0.05 (0.12)	0.44 (0.05)	0.52 (0.02)	2.95 (1.04)	0.26 (0.07)
STaSy	3.41 (1.39)	3.66 (1.42)	0.38 (0.05)	0.63 (0.05)	-4.21 (4.44)	0.61 (0.06)	0.46 (0.02)	1.23 (0.44)	0.45 (0.12)
TabDDPM	4.27 (1.89)	4.79 (1.89)	0.76 (0.06)	0.80 (0.06)	0.60 (0.11)	0.66 (0.06)	0.39 (0.03)	0.76 (0.28)	0.72 (0.11)
Forest-VP	1.46 (0.40)	1.94 (0.50)	0.67 (0.05)	0.84 (0.03)	0.55 (0.10)	0.73 (0.04)	0.39 (0.01)	0.94 (0.30)	0.52 (0.15)
Forest-Flow	1.36 (0.39)	1.90 (0.5)	0.83 (0.03)	0.90 (0.03)	0.57 (0.11)	0.73 (0.04)	0.38 (0.01)	0.83 (0.23)	0.63 (0.11)
Oracle	0.00 (0.00)	1.81 (0.47)	0.99 (0.01)	0.91 (0.04)	0.64 (0.09)	0.77 (0.04)	NA	0.00 (0.00)	1.00 (0.00)

Table 9: Tabular data imputation (27 datasets, 3 experiments per dataset, 10 imputations per experiment) with 20% missing values; raw score - mean (standard-error)

	MinMAE ↓	AvgMAE ↓	$W_{train}\downarrow$	$W_{test}\downarrow$	MAD ↑	$R^2_{imp}\uparrow$	$F1_{imp}\uparrow$	$P_{bias}\downarrow$	$Cov_{rate}\uparrow$
KNN	0.16 (0.03)	0.16 (0.03)	0.42 (0.08)	1.89 (0.49)	0.00 (0.00)	0.59 (0.09)	0.75 (0.04)	1.27 (0.25)	0.40 (0.11)
ICE	0.10 (0.01)	0.21 (0.03)	0.52 (0.09)	1.99 (0.49)	0.69 (0.10)	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 (0.10)	0.76 (0.04)	0.61 (0.20)	0.75 (0.11)
MissForest	0.10 (0.03)	0.12 (0.03)	0.32 (0.07)	1.85 (0.48)	0.10 (0.03)	0.61 (0.10)	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)	0.00 (0.00)	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 (0.10)	1.98 (0.49)	0.28 (0.05)	0.59 (0.10)	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 (0.00)	0.60 (0.10)	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 (0.50)	0.25 (0.03)	0.61 (0.10)	0.74 (0.04)	0.81 (0.25)	0.57 (0.14)
Oracle	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1.87 (0.49)	0.00 (0.00)	0.64 (0.09)	0.78 (0.04)	0.00 (0.00)	1.00 (0.00)

Table 10: 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 and Forest-Flow; raw score - mean (standard-error)

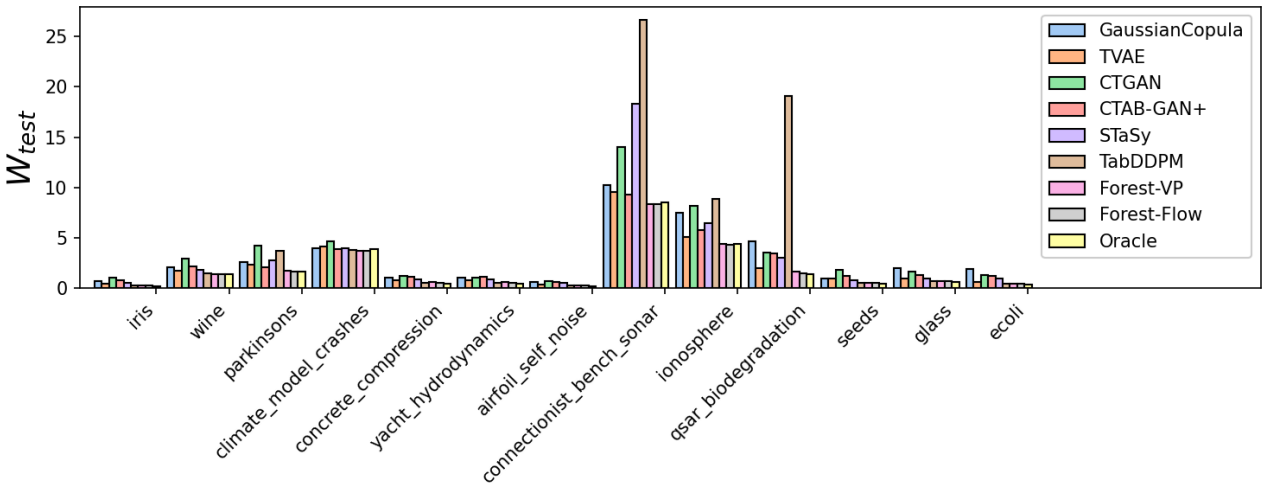
	$W_{train}\downarrow$	$W_{test}\downarrow$	$cov_{train}\uparrow$	$cov_{test}\uparrow$	$R^2_{fake}\uparrow$	$F1_{fake}\uparrow$	$F1_{disc}\uparrow$	$P_{bias}\downarrow$	$cov_{rate}\uparrow$
GaussianCopula	2.60 (0.58)	2.86 (0.63)	0.20 (0.04)	0.43 (0.05)	0.20 (0.18)	0.48 (0.06)	0.60 (0.04)	2.31 (1.00)	0.21 (0.08)
TVAE	2.17 (0.60)	2.40 (0.65)	0.32 (0.04)	0.63 (0.04)	-0.66 (0.95)	0.55 (0.08)	0.45 (0.01)	4.04 (2.30)	0.29 (0.09)
CTGAN	3.64 (1.02)	3.79 (1.04)	0.12 (0.03)	0.28 (0.05)	-0.34 (0.14)	0.36 (0.04)	0.48 (0.01)	2.47 (1.24)	0.20 (0.06)
CTABGAN	2.76 (0.83)	2.95 (0.86)	0.23 (0.04)	0.45 (0.05)	0.08 (0.12)	0.45 (0.05)	0.50 (0.02)	2.10 (0.58)	0.25 (0.06)
Stasy	3.40 (1.37)	3.67 (1.4)	0.38 (0.05)	0.63 (0.06)	0.27 (0.28)	0.64 (0.06)	0.46 (0.02)	1.09 (0.22)	0.36 (0.10)
TabDDPM	4.36 (1.89)	4.80 (1.90)	0.72 (0.06)	0.78 (0.06)	0.58 (0.11)	0.67 (0.06)	0.42 (0.03)	1.16 (0.35)	0.56 (0.10)
Forest-VP	1.84 (0.51)	2.14 (0.56)	0.53 (0.04)	0.78 (0.03)	0.53 (0.10)	0.71 (0.04)	0.42 (0.01)	1.16 (0.30)	0.43 (0.12)
Forest-Flow	1.82 (0.51)	2.12 (0.56)	0.67 (0.03)	0.84 (0.03)	0.55 (0.11)	0.69 (0.04)	0.43 (0.01)	1.16 (0.32)	0.50 (0.10)
Oracle	0.00 (0.00)	1.87 (0.49)	0.99 (0.01)	0.91 (0.04)	0.64 (0.09)	0.78 (0.04)	NA	0.00 (0.00)	1.00 (0.00)

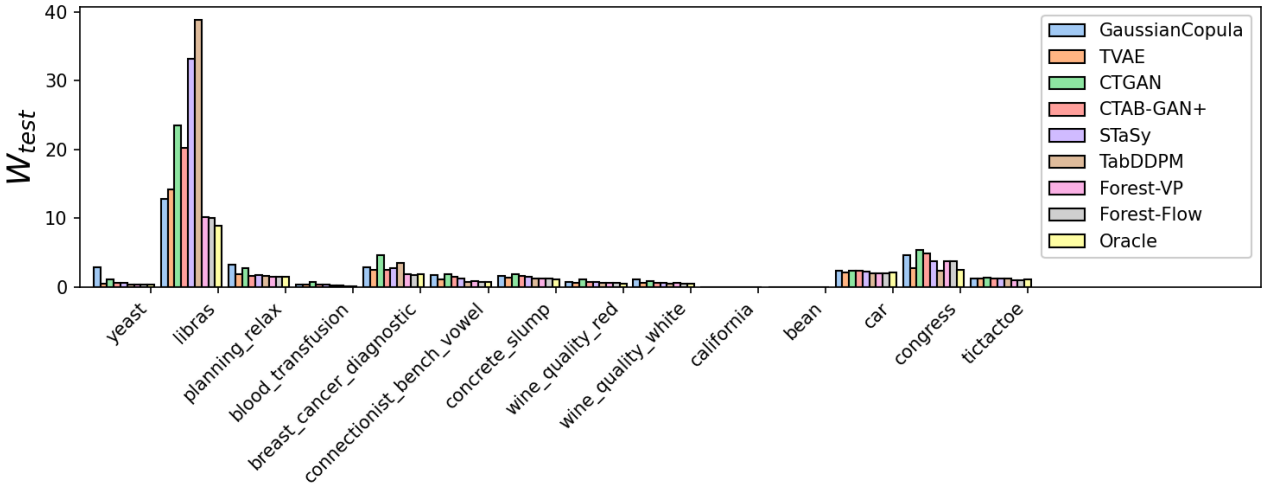
C.2 Bar plots

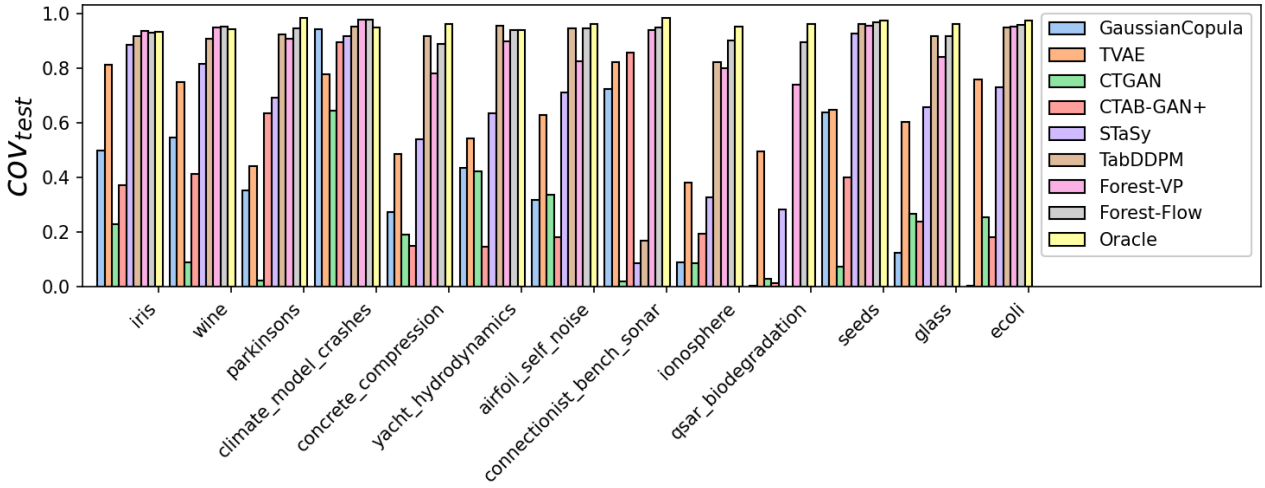
In this section, we provide the bar plots for each method for all datasets and all metrics for the generation of tabular data C.2.1, the tabular data imputation C.2.2 and the tabular data generation with missing values C.2.3.

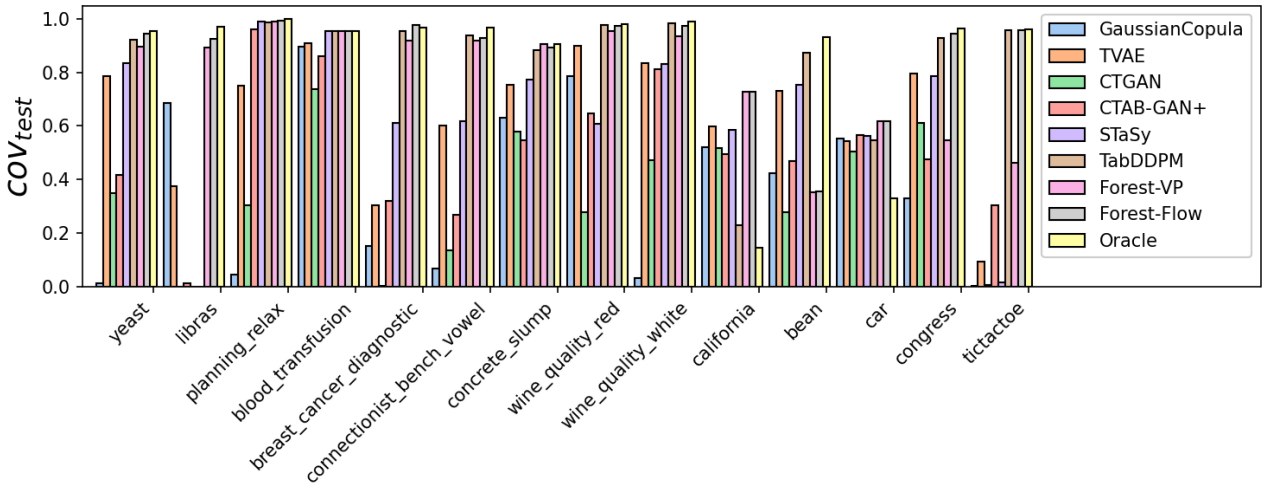
Please ignore the $F1_{disc}$ results for Oracle; there is not proper way to train and test on different training data, so the values are incorrect.

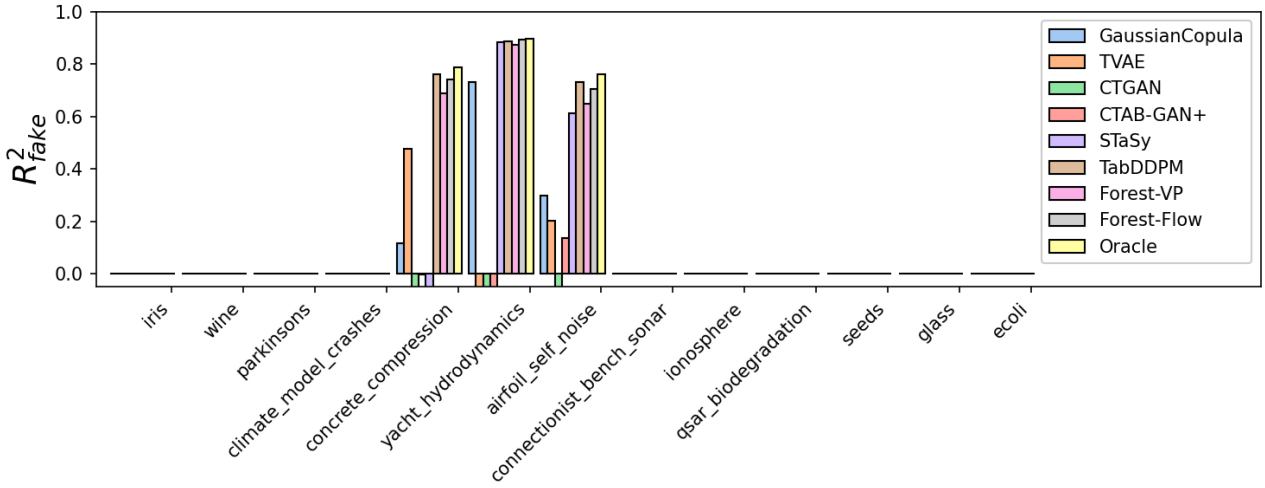
C.2.1 Bar plots for generation with complete data

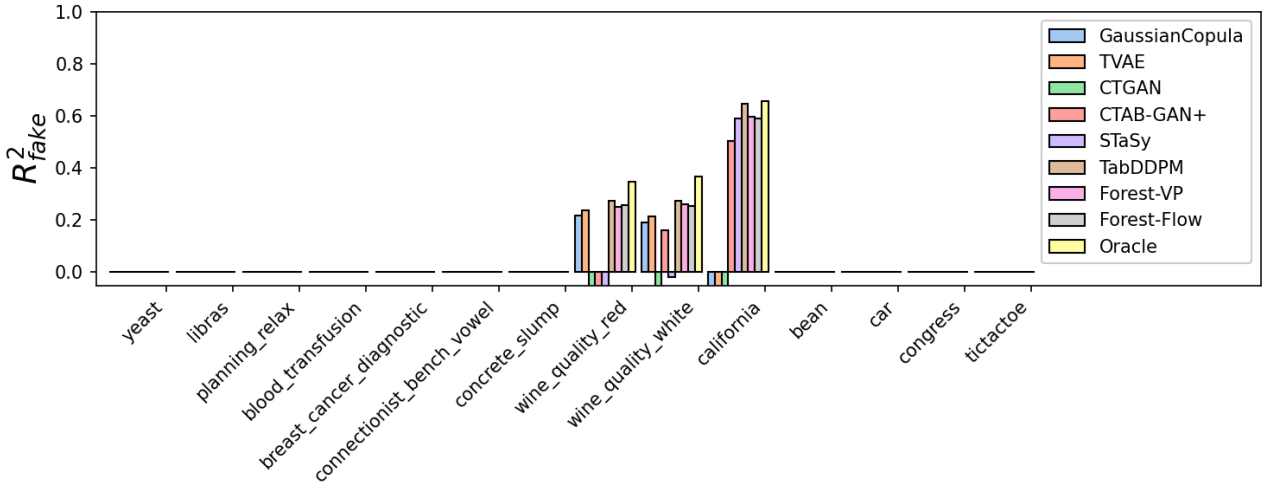


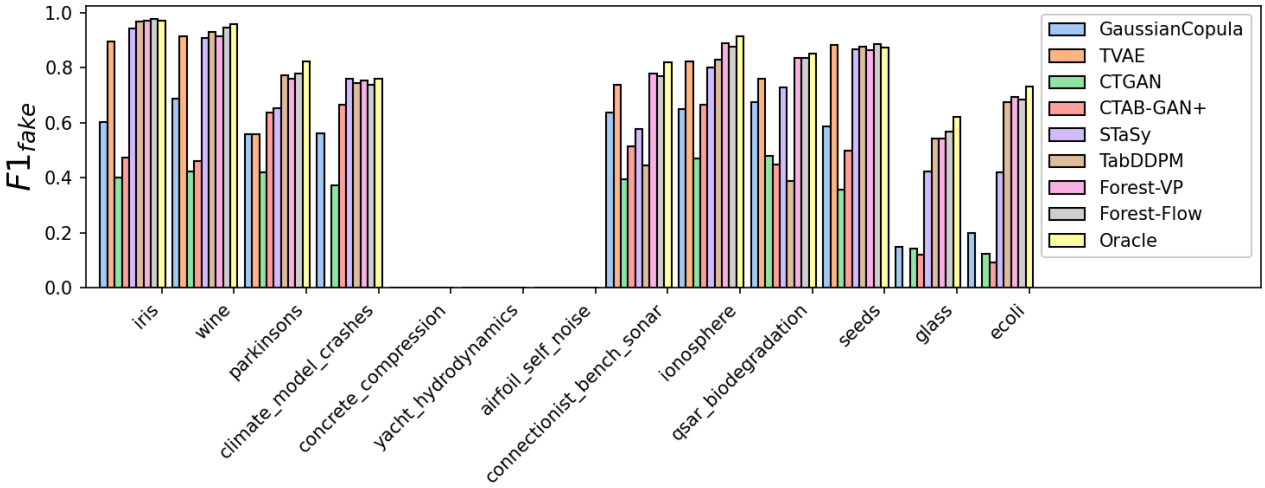


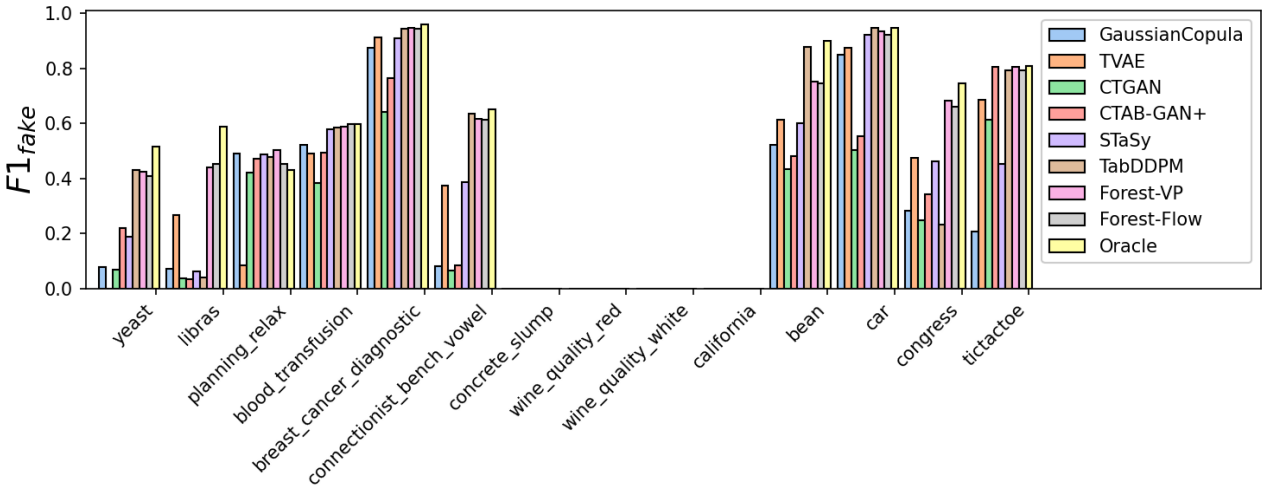

COV_{test}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
ecoli

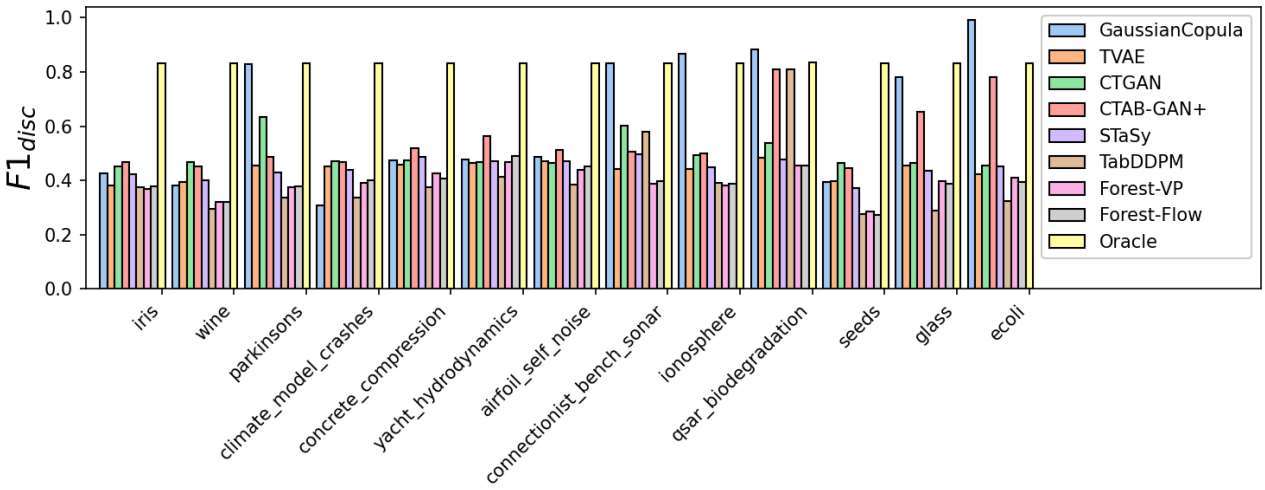

COV_{test}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
yeast
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

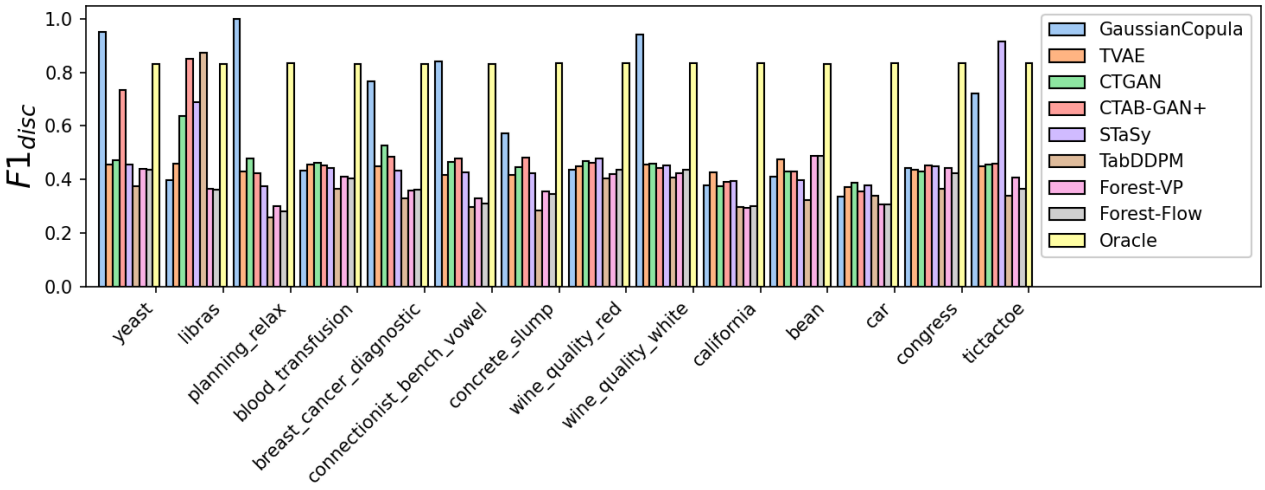

R^2_{fake}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
ecoli

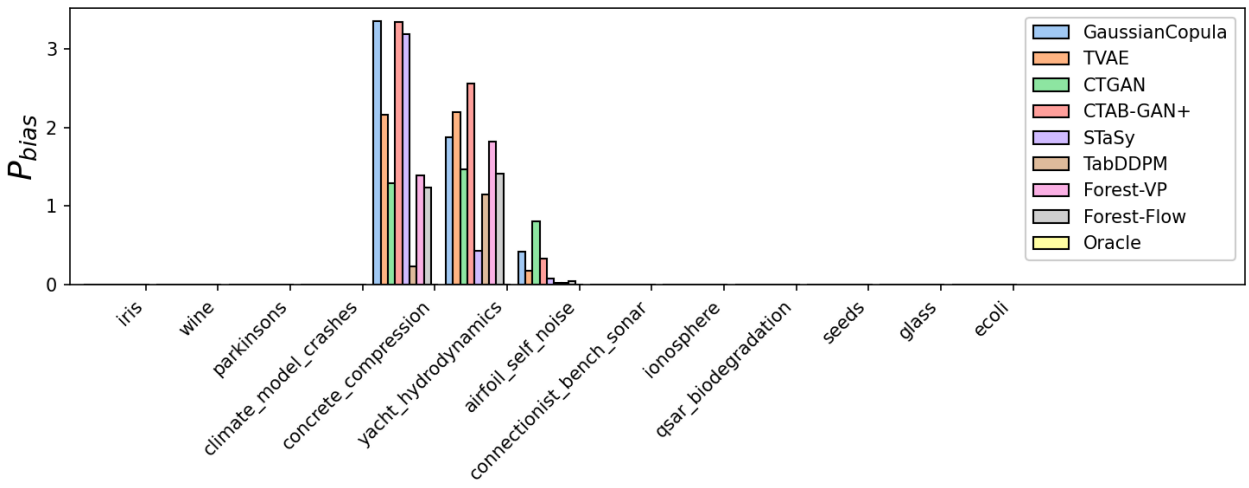

R^2_{fake}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
yeast
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

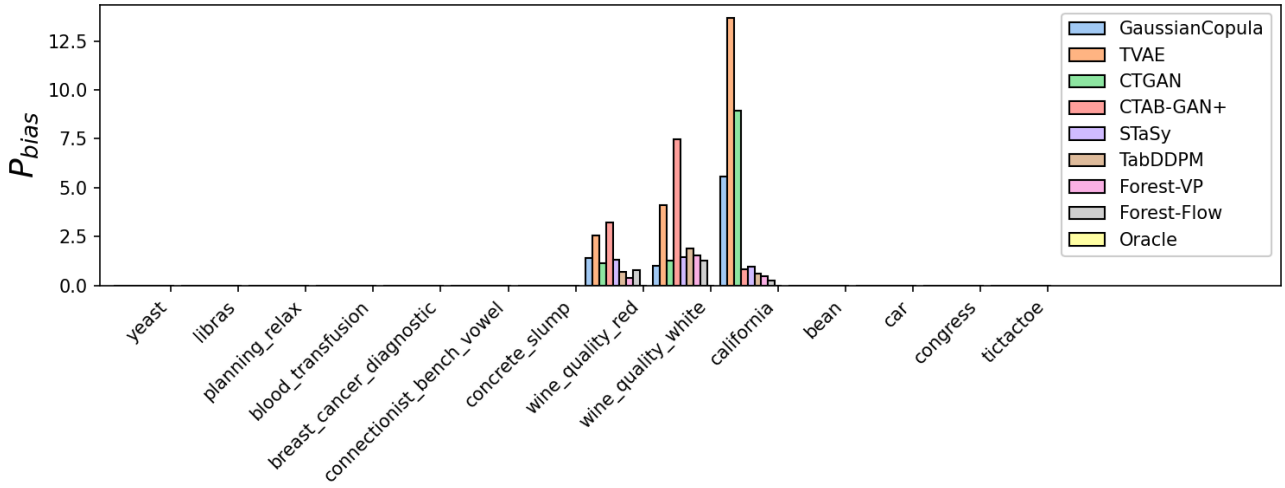

$F1_{fake}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
ecoli

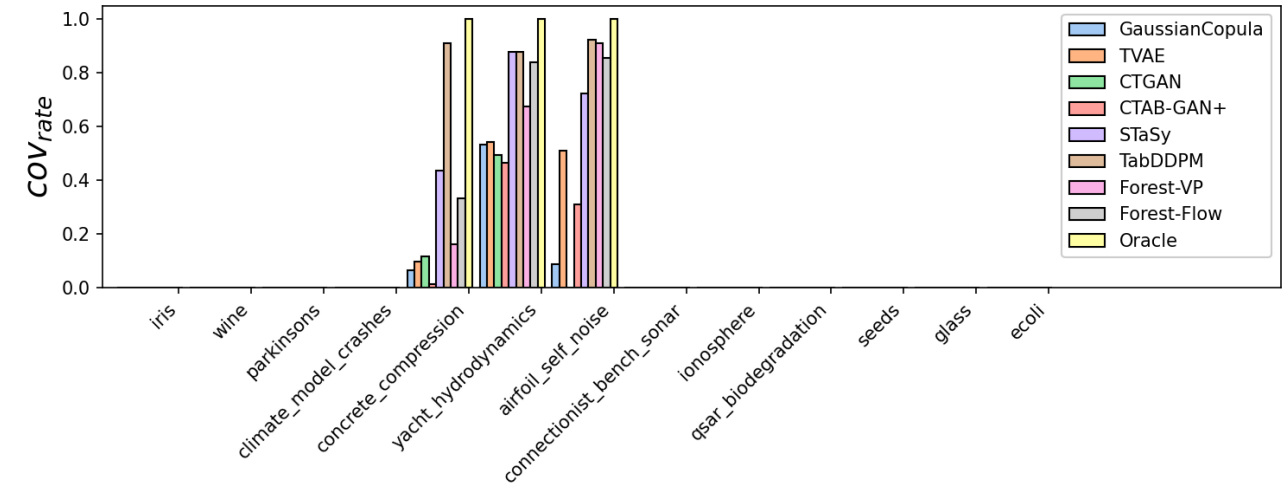

$F1_{fake}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
yeast
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

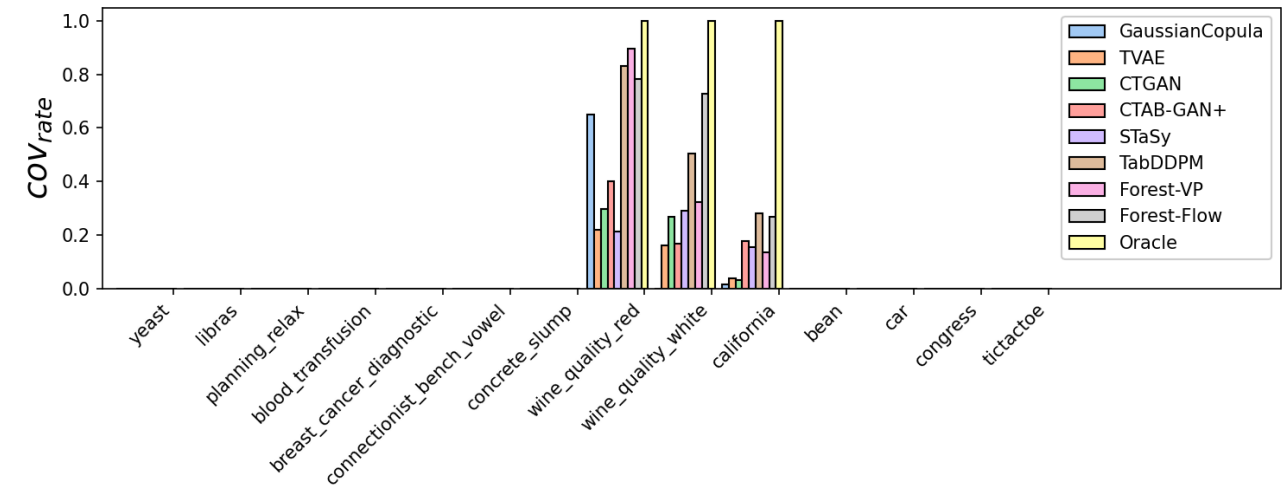

$F1_{disc}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
ecoli


$F1_{disc}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
yeast
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

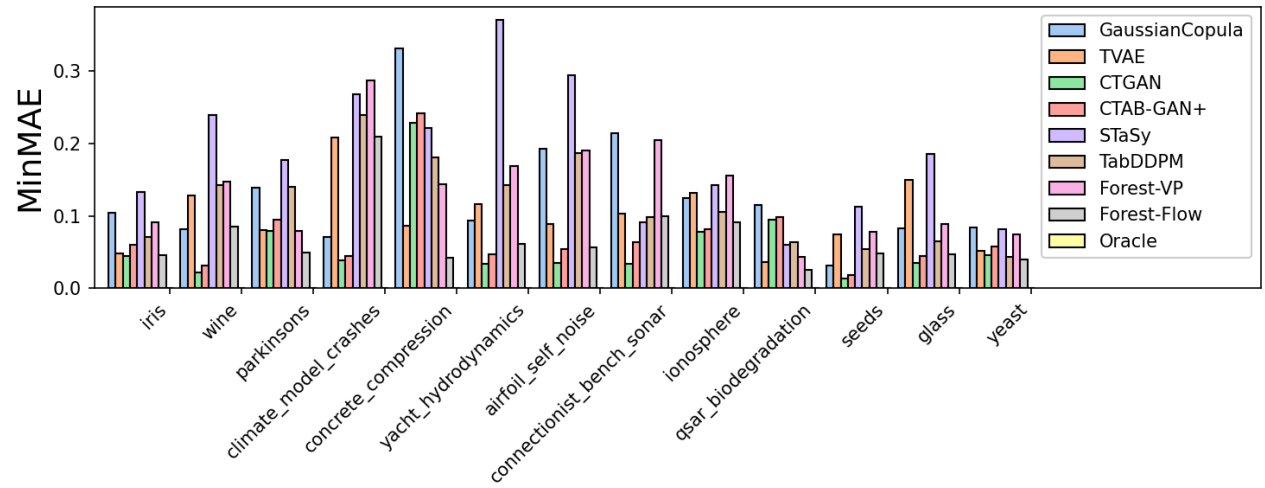

P_{bias}
3
2
1
0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
ecoli

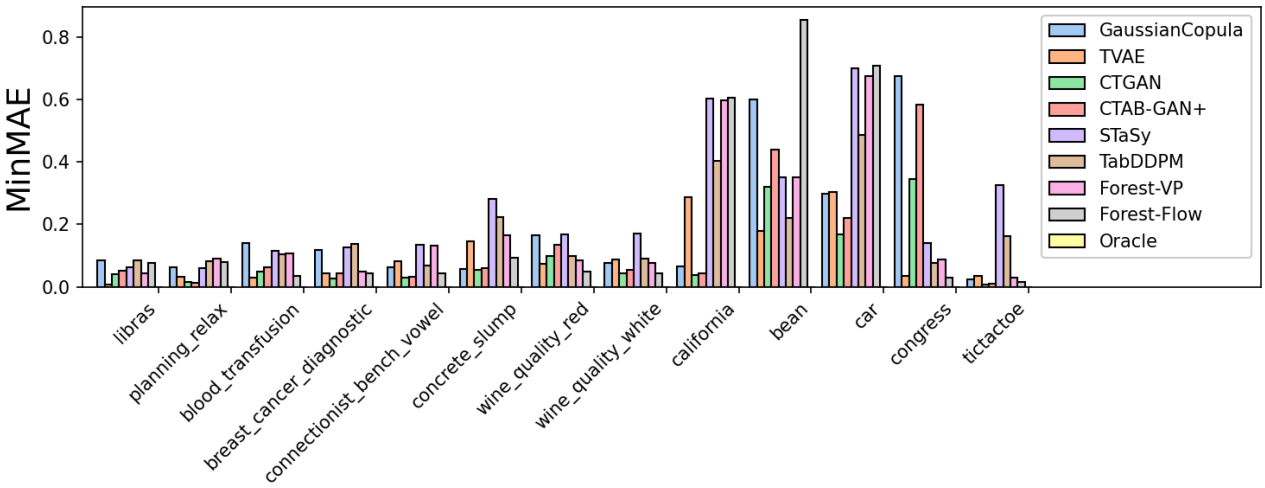

P_{bias}
0.0
2.5
5.0
7.5
10.0
12.5
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
yeast
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

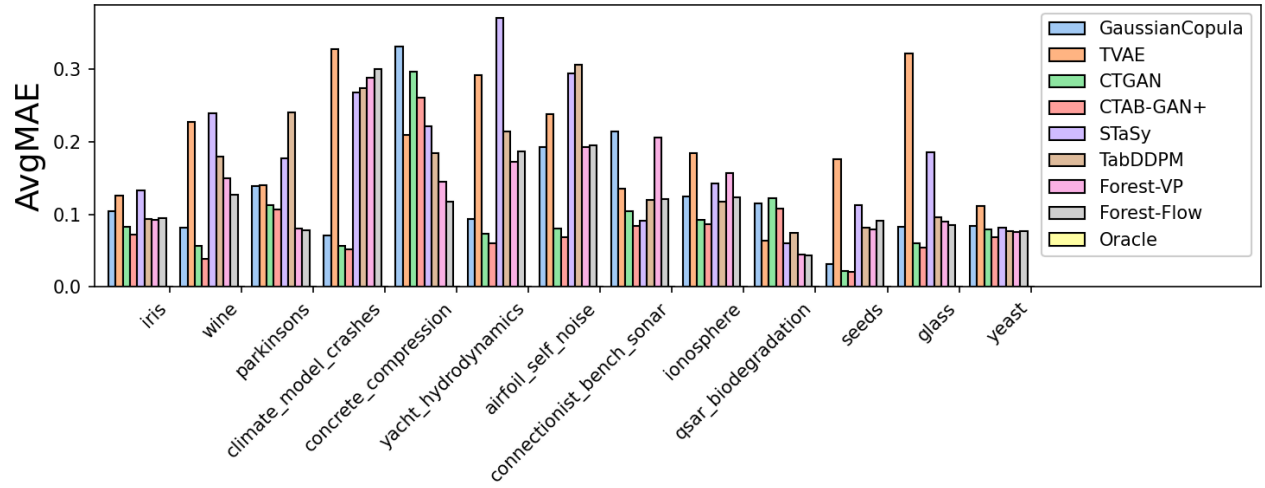

COV_{rate}
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
ecoli

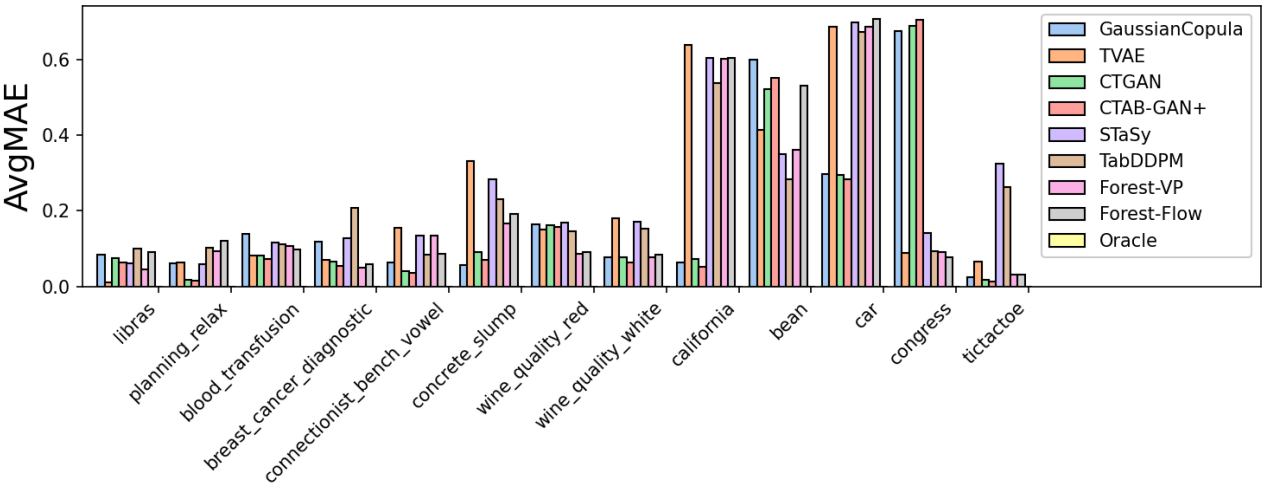

COV_{rate}
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
yeast
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

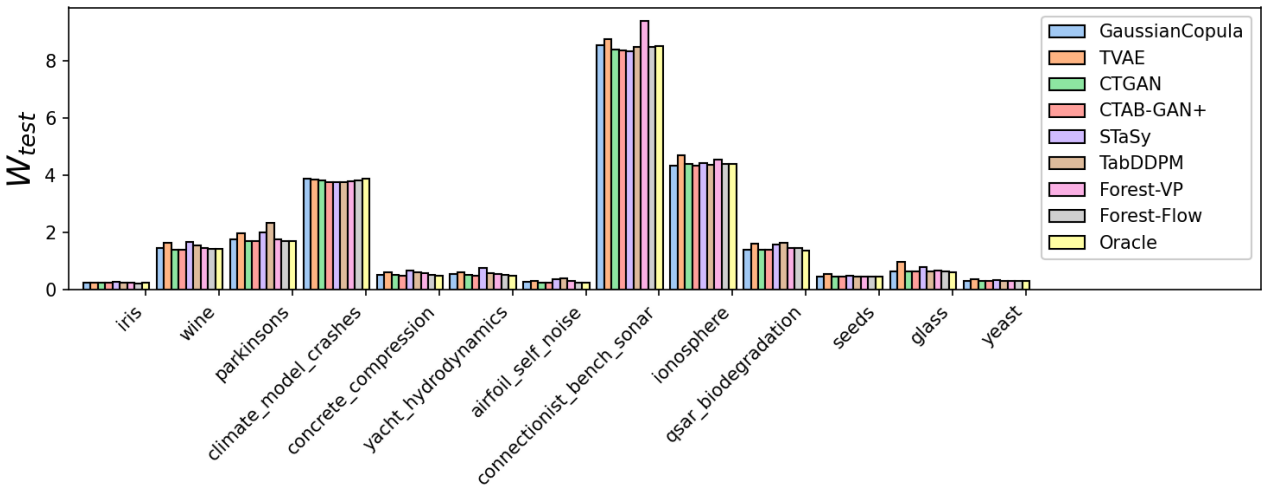
C.2.2 Bar plots for imputation

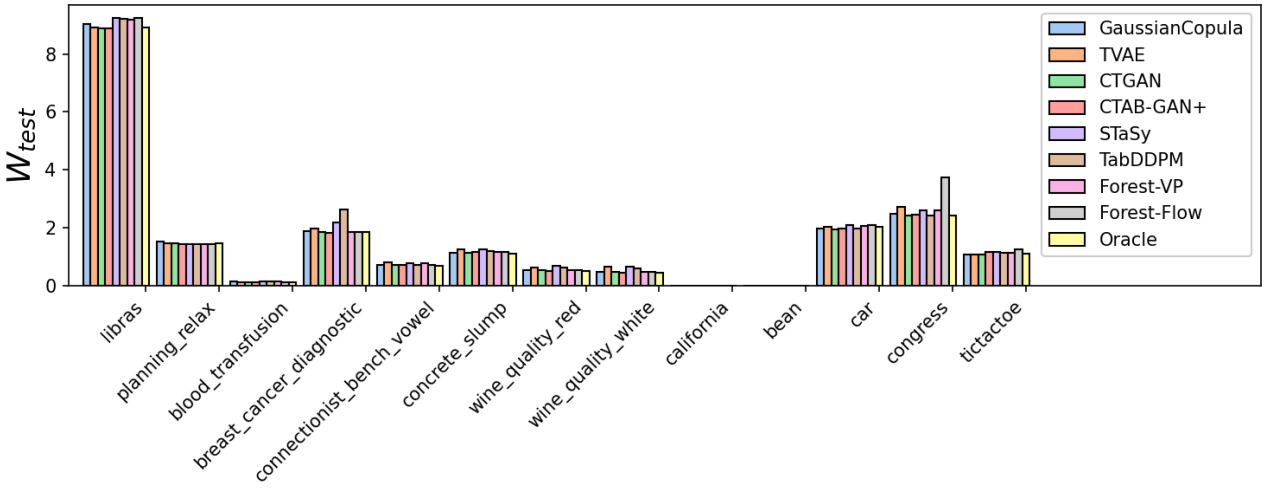


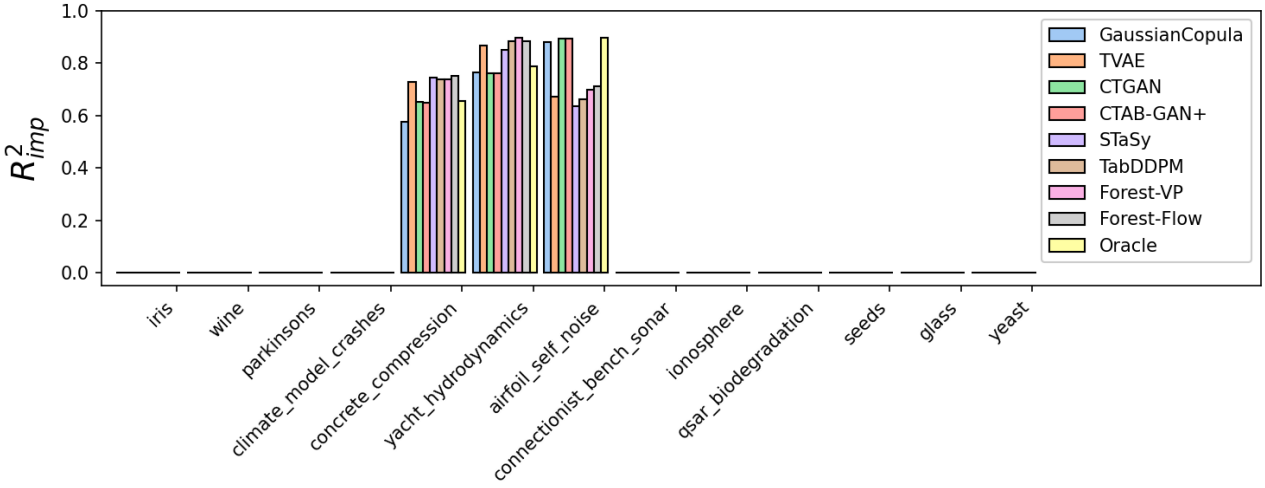


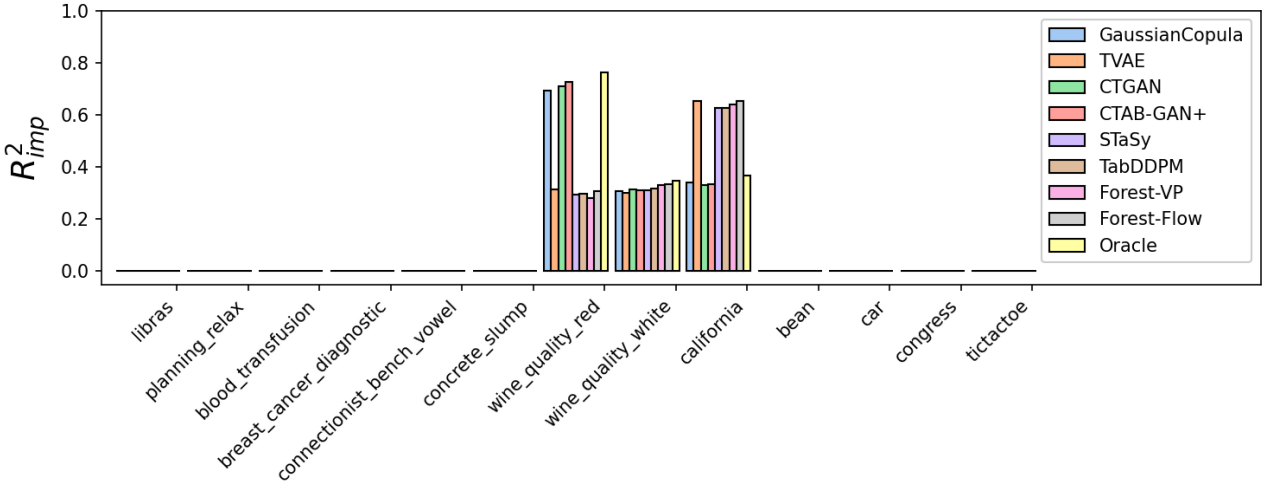

AvgMAE
0.0
0.1
0.2
0.3
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

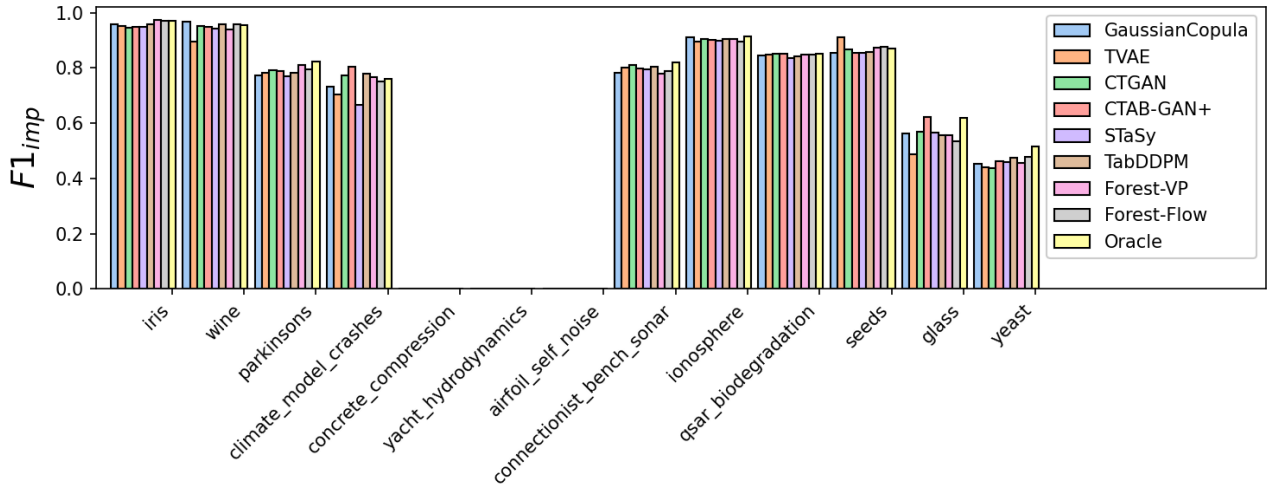

AvgMAE
0.0
0.2
0.4
0.6
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

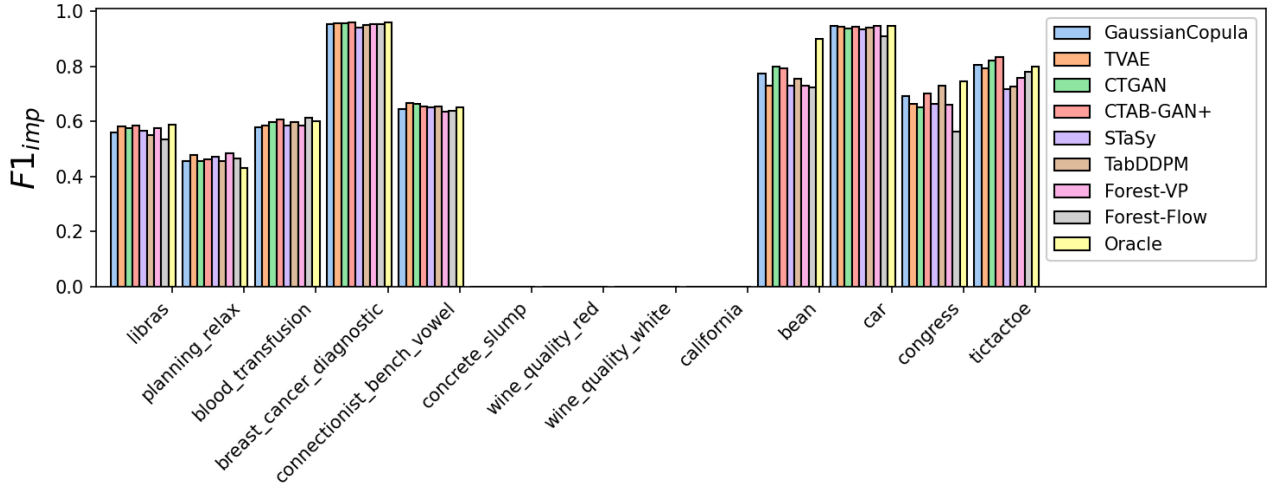

W_{test}
0
2
4
6
8
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

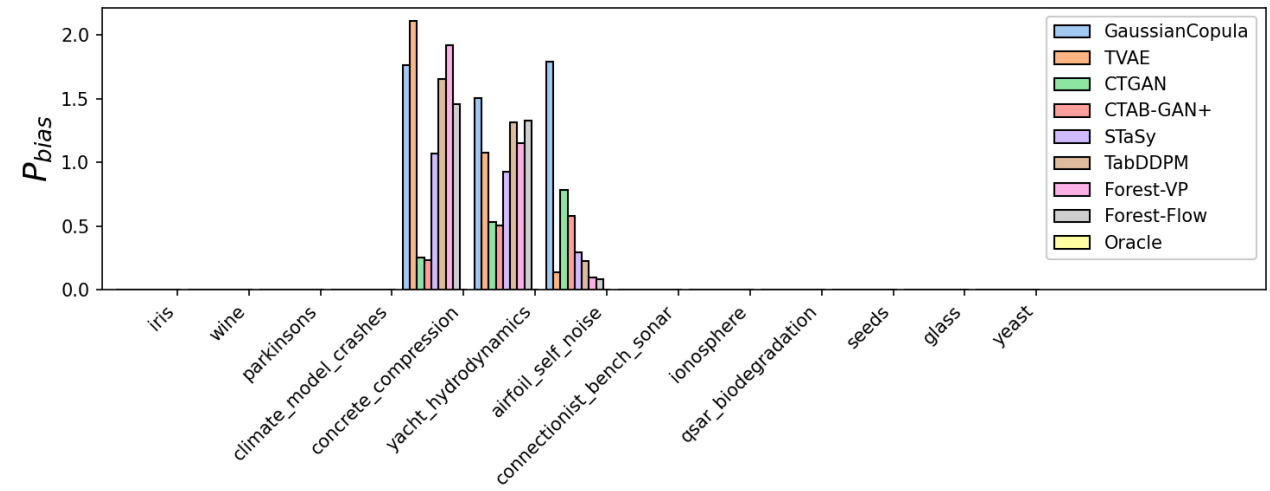

W_{test}
0
2
4
6
8
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

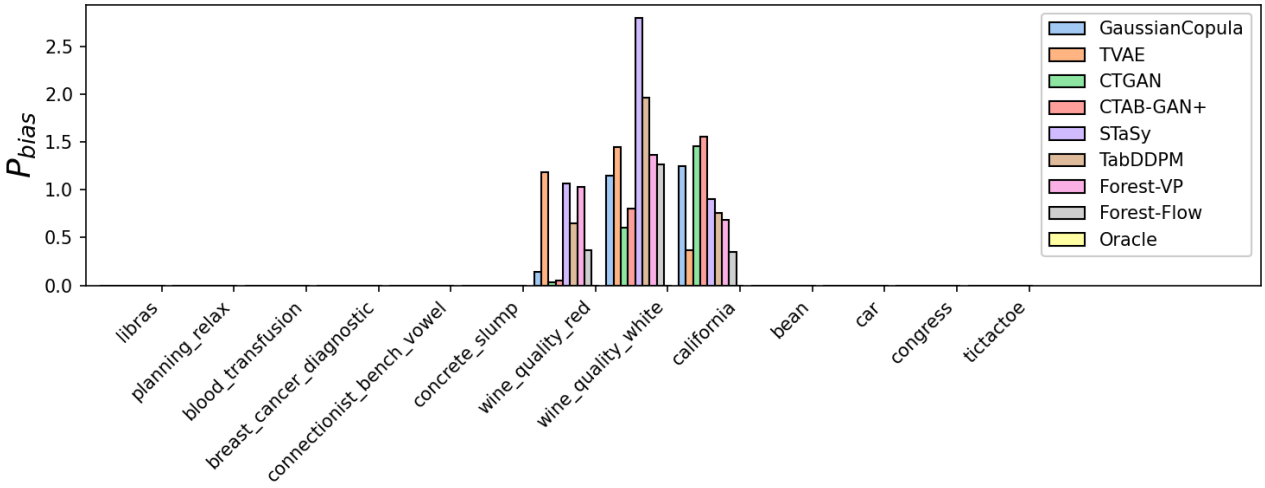

R^2_{imp}
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

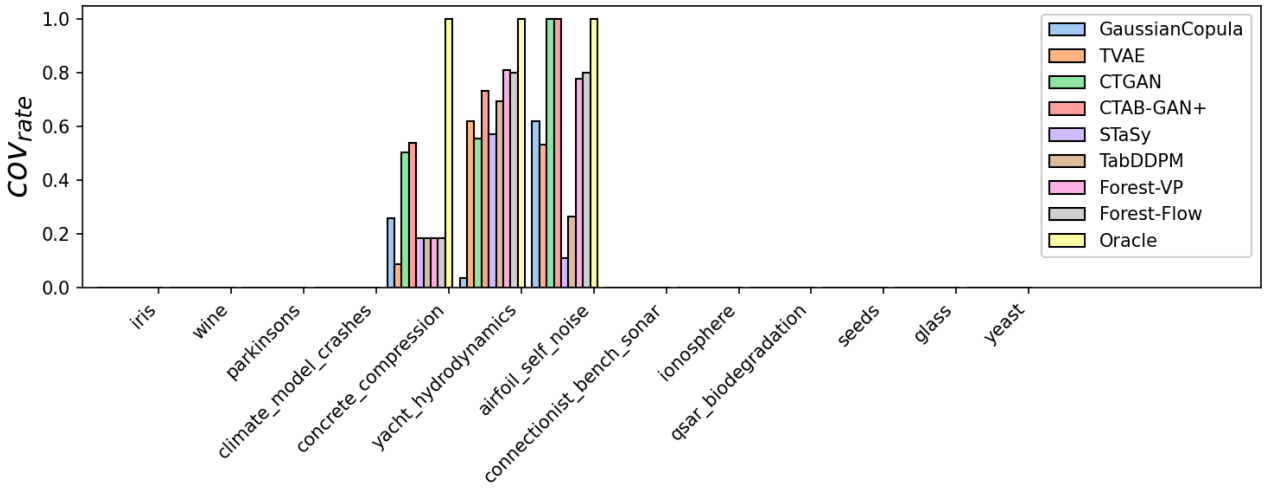

R^2_{imp}
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

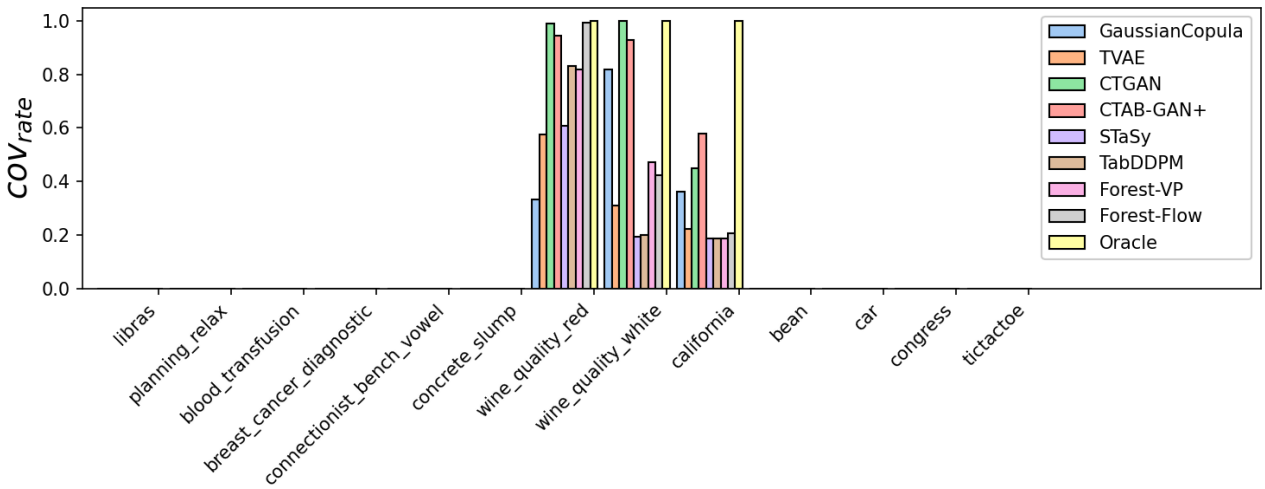

$F1_{imp}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast


$F1_{imp}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

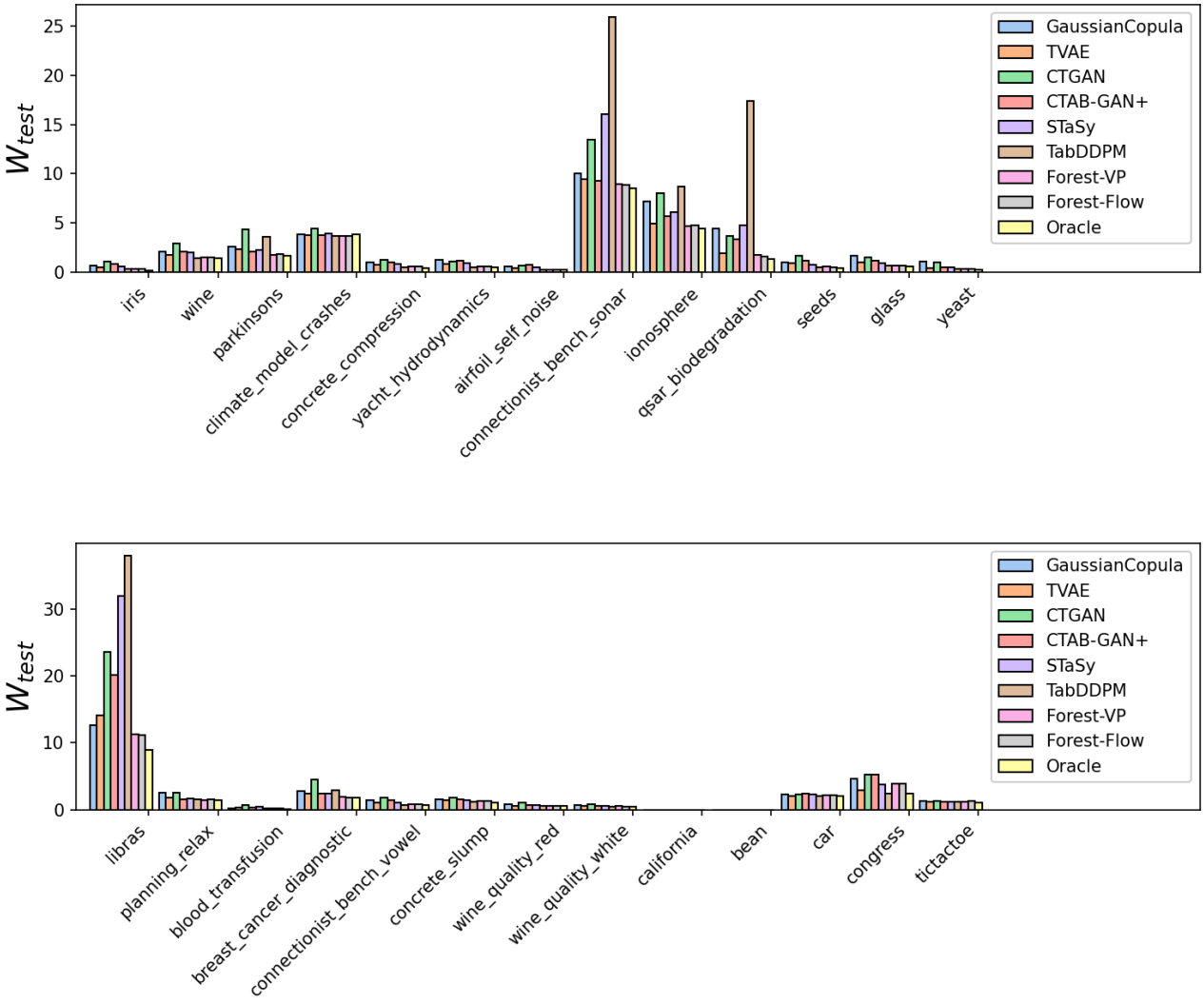

P_{bias}
2.0
1.5
1.0
0.5
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

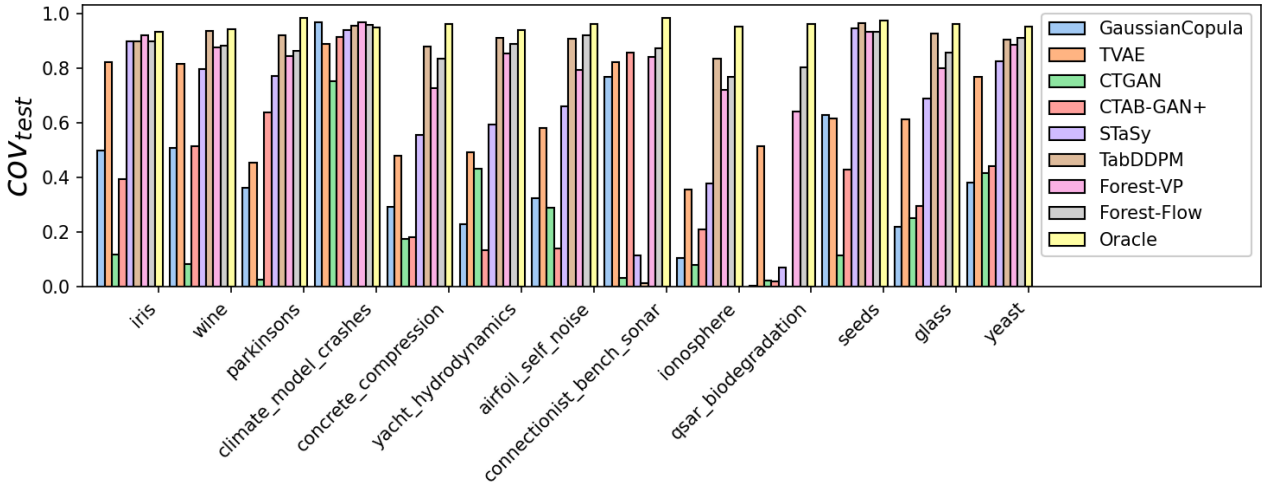

P_{bias}
0.0
0.5
1.0
1.5
2.0
2.5
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

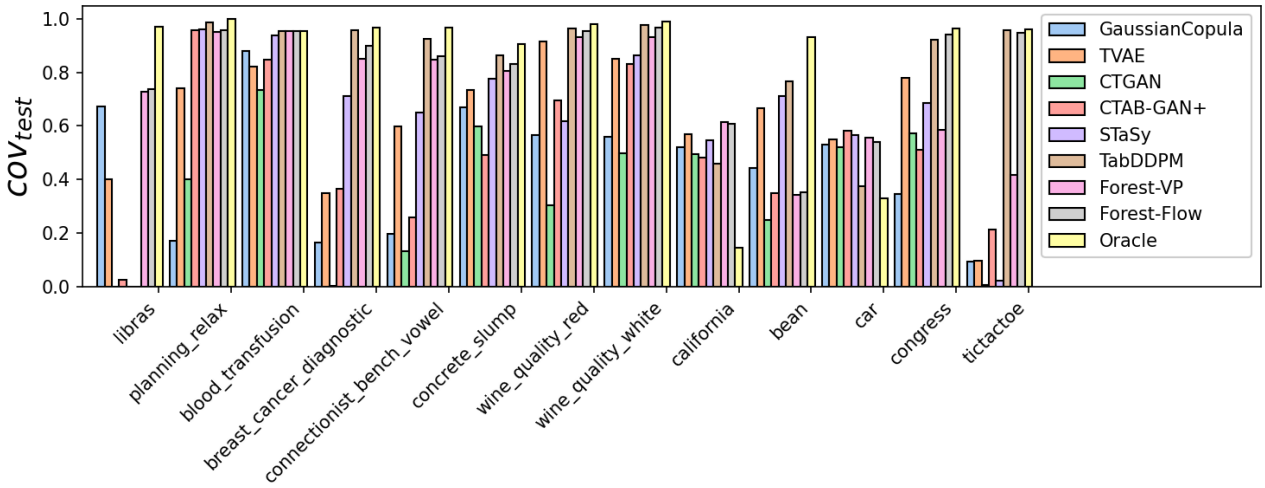

COV_{rate}
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

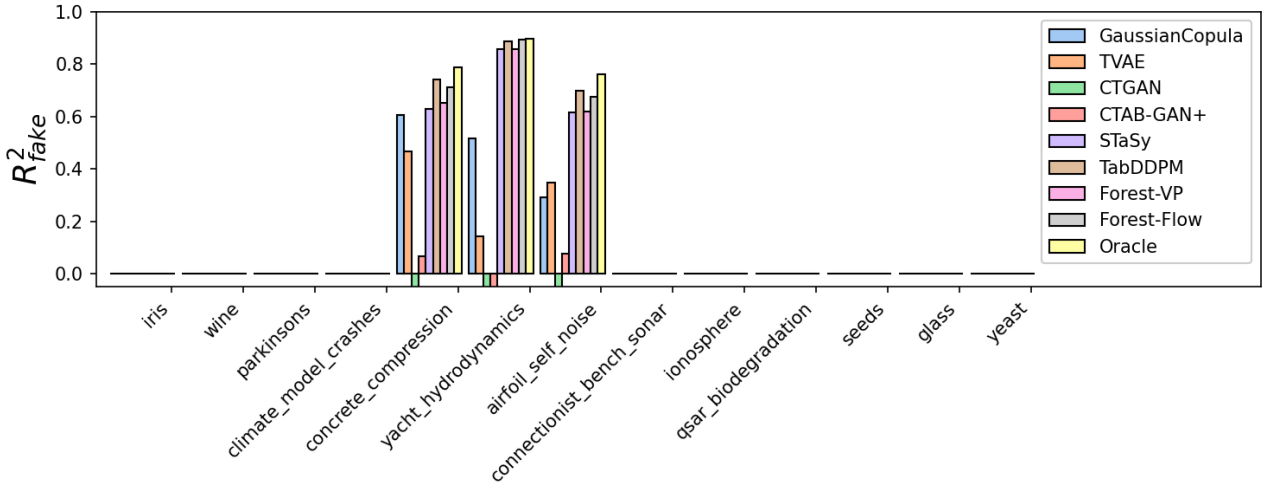

COV_{rate}
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

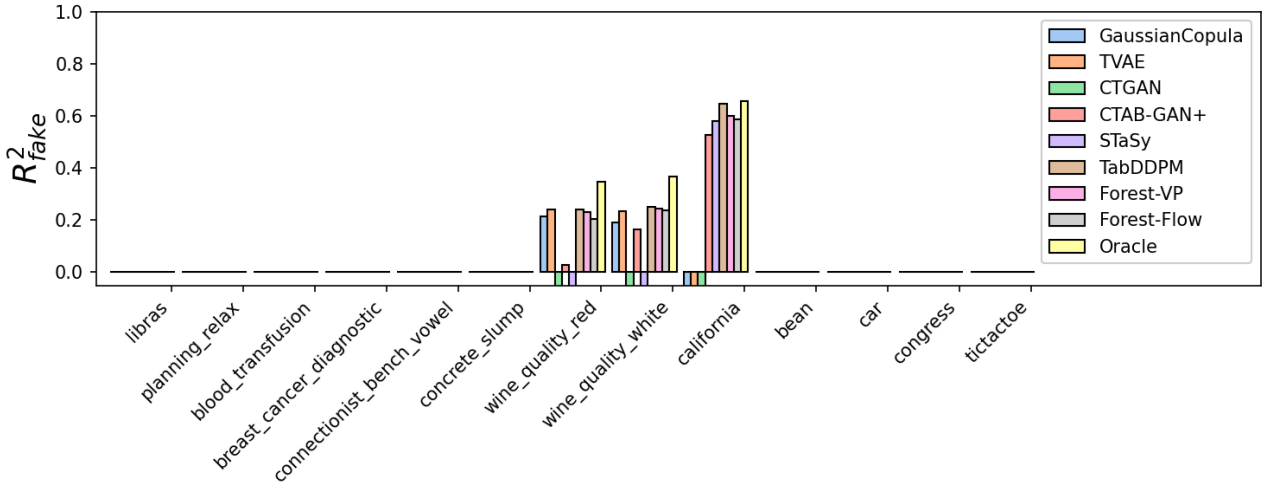
C.2.3 Bar plots for generation with incomplete data

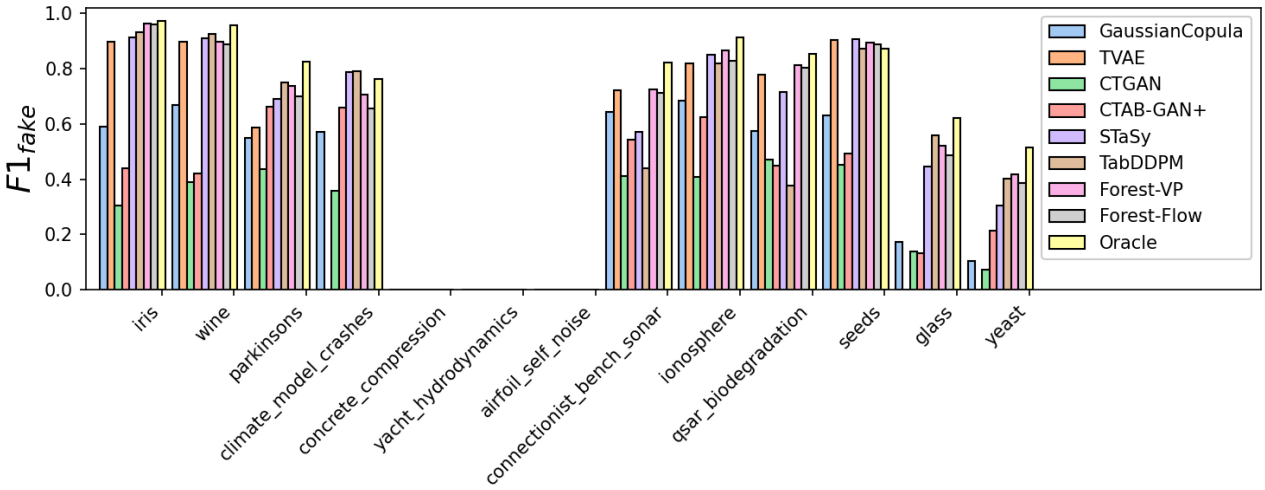


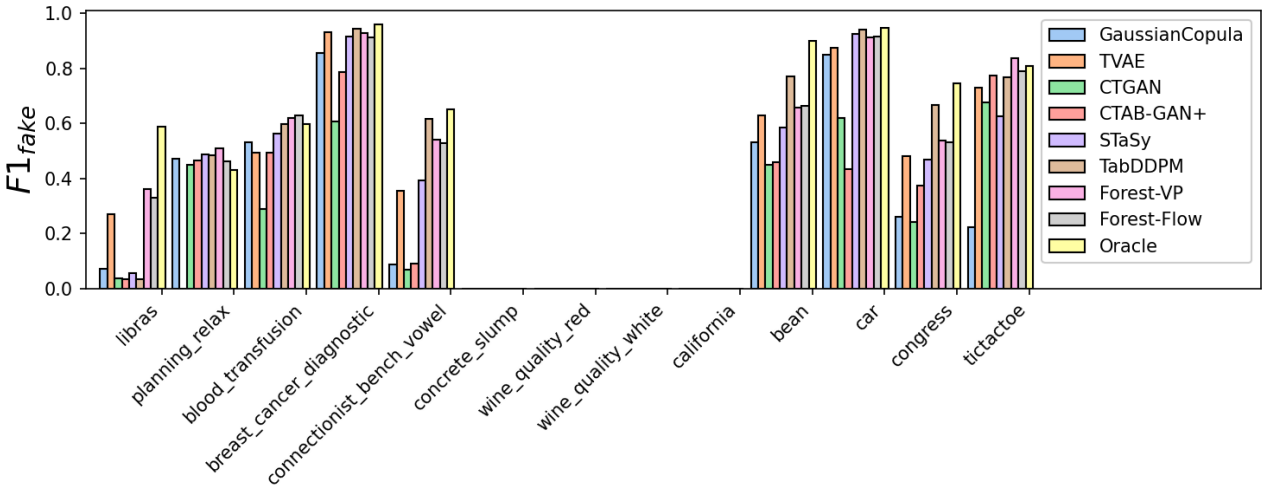

COV_{test}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

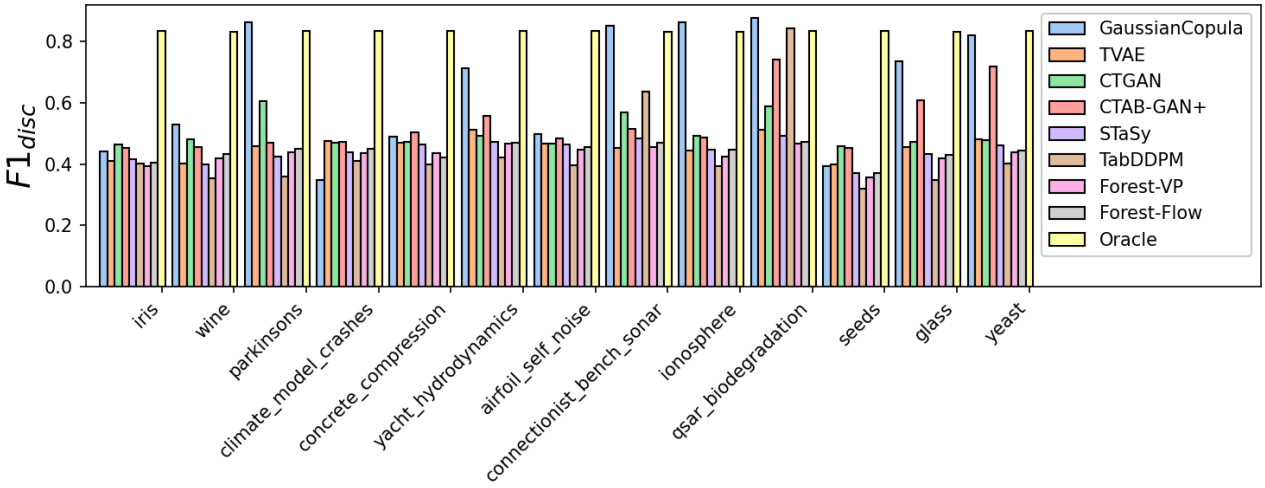

COV_{test}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

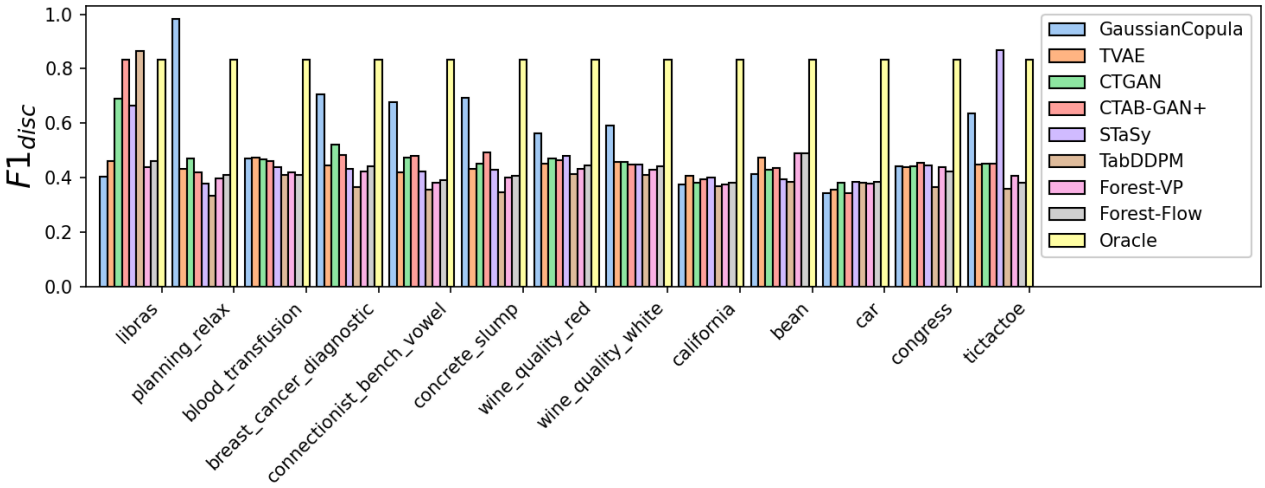

R^2_{fake}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

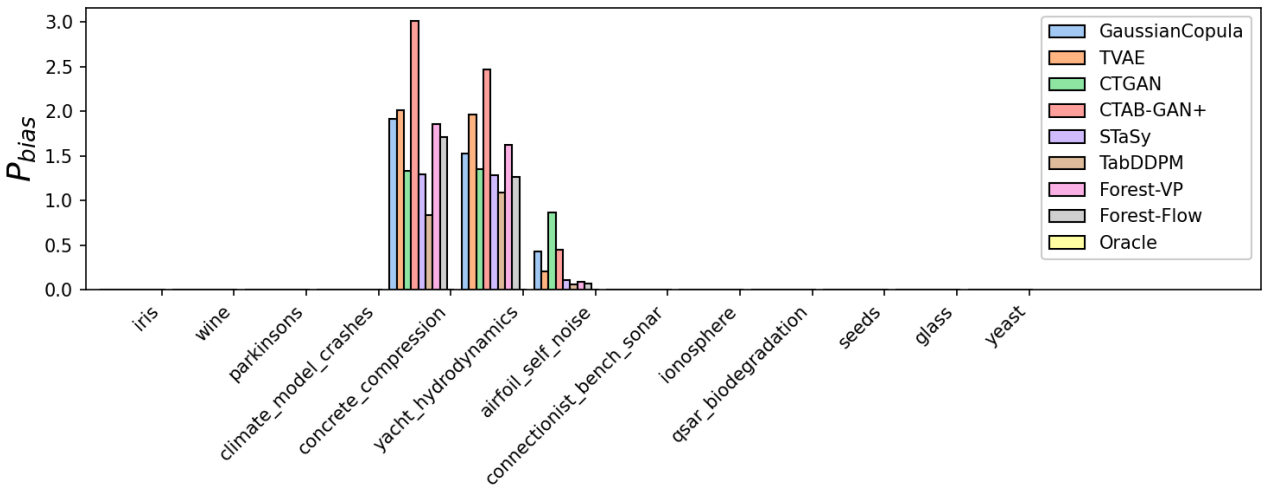

R^2_{fake}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

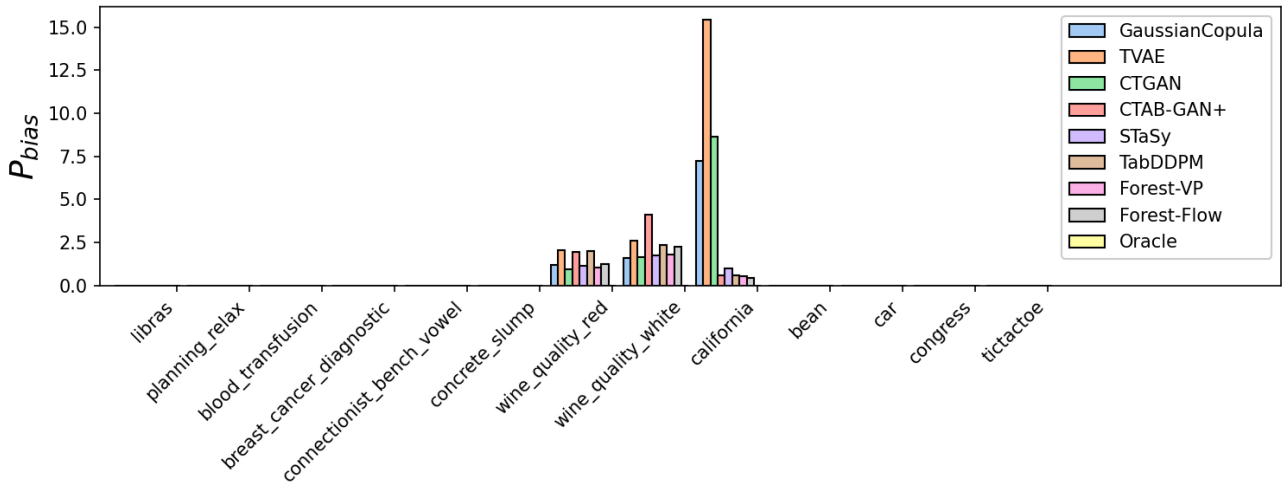

$F1_{fake}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast

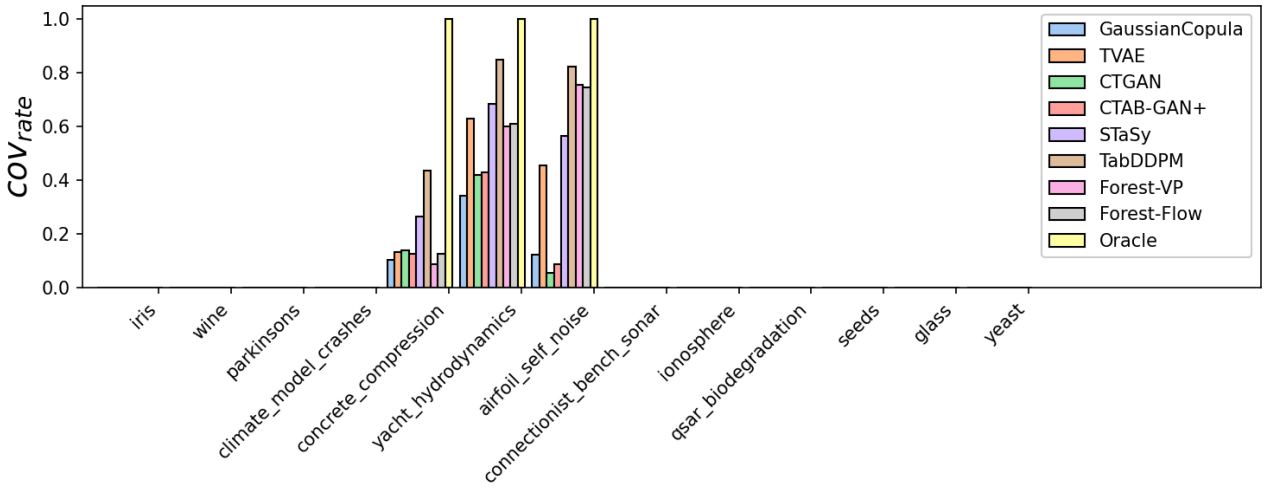

$F1_{fake}$
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

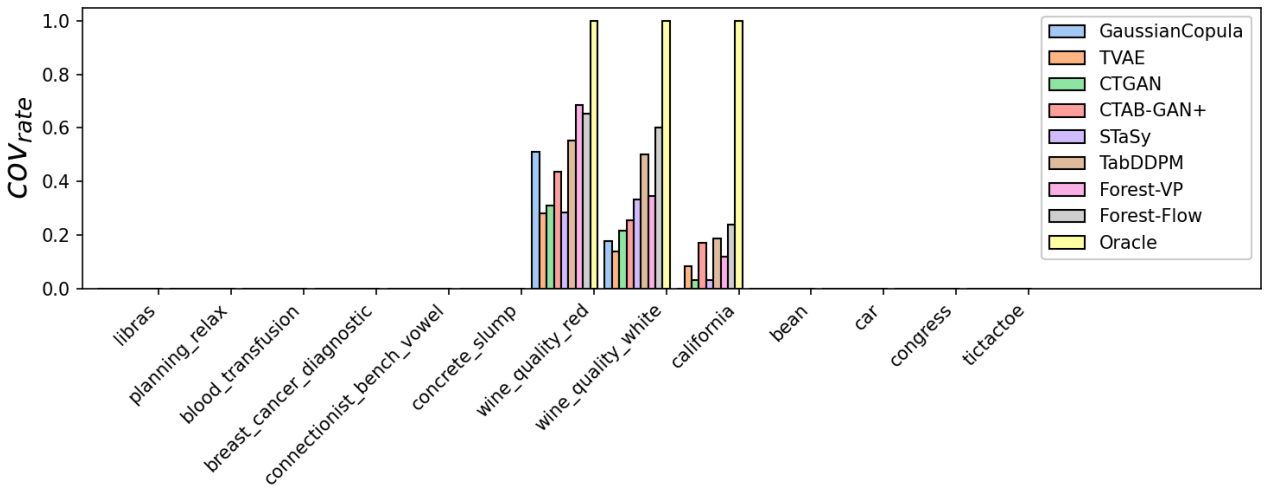

$F1_{disc}$
0.0
0.2
0.4
0.6
0.8
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast


$F1_{disc}$
0.0
0.2
0.4
0.6
0.8
1.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe


P_{bias}
0.0
0.5
1.0
1.5
2.0
2.5
3.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast


P_{bias}
15.0
12.5
10.0
7.5
5.0
2.5
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe


COV_{rate}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
iris
wine
parkinsons
climate_model_crashes
concrete_compression
yacht_hydrodynamics
airfoil_self_noise
connectionist_bench_sonar
ionosphere
qsar_biodegradation
seeds
glass
yeast


COV_{rate}
1.0
0.8
0.6
0.4
0.2
0.0
GaussianCopula
TVAE
CTGAN
CTAB-GAN+
STaSy
TabDDPM
Forest-VP
Forest-Flow
Oracle
libras
planning_relax
blood_transfusion
breast_cancer_diagnostic
connectionist_bench_vowel
concrete_slump
wine_quality_red
wine_quality_white
california
bean
car
congress
tictactoe

C.3 UMAP

In this section, we show a Uniform Manifold Approximation (UMAP) (McInnes et al., 2018) visualization of the red wine quality dataset comparing oracle, ForestFlow (ours), and TabDDPM.