Knowledge-Based Systems 280 (2023) 110956

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

GenerativeMTD: A deep synthetic data generation framework for small datasets

Jayanth Sivakumar [a], Karthik Ramamurthy [b], Menaka Radhakrishnan [b], Daehan Won [a,*]

[a] Department of Systems Science and Industrial Engineering, Binghamton University, P.O. Box 6000, Binghamton, NY 13902-6000, USA
[b] Center for Cyber Physical Systems (CCPS), Vellore Institute of Technology, Chennai, Tamil Nadu, 600 127, India

ARTICLE INFO

Article history:
Received 15 March 2023
Received in revised form 28 August 2023
Accepted 30 August 2023
Available online 4 September 2023

Dataset link: https://github.com/jsivaku1/GenerativeMTD

Keywords:
Small dataset
Synthetic data generation
Deep learning
Privacy-preserving

ABSTRACT

Synthetic data generation for tabular data unlike computer vision, is an emerging challenge. When tabular data needs to be synthesized, it either faces a small dataset problem or violates privacy if the data contains sensitive information. When the data is small, any data-driven modeling leads to biased decision making. On the other hand, deep learning models that use small dataset for training are limited. Tabular data also faces a myriad of challenges, such as mixed data types, fidelity, mode collapse, etc. To eradicate small dataset problems and increase the deep learning capabilities on small data, a new generative method, GenerativeMTD, is proposed in this research. The method generates fake data by using pseudo-real data as input during the training. Pseudo-real data serves the purpose of training the deep learning model with large samples when the real dataset size is small. The pseudo-real data is generated from the real data through *k*-nearest neighbor mega-trend diffusion. This pseudo-real data is then translated into synthetic data that is similar and realistic to the real data. The method outperforms some of the state-of-the-art methodologies that exist in tabular data generation. The proposed method also generates quality synthetic data for the benchmark datasets in terms of pairwise correlation differences. In addition, the method surpasses the benchmark models in terms of the distance-based privacy metrics: distance to the closest record and nearest neighbor distance ratio.

1. Introduction

Data availability is a fundamental step for any data-driven modeling. When the data is available, it is either small or may violate privacy due to personal identifiable information (PII). A rising pandemic or a new product trial that has a limited amount of data requires sophisticated approaches for successful decision making. A promising alternative to tackle data availability is synthetic data. Irrespective of the type of data, synthetic data is a propitious surrogate to replace the real data. In addition, when the data is not accessible due to privacy constraints or when the dataset is small, real data can be replaced with realistic and privacy-preserving synthetic data for research and analysis.

Synthetic data generation for tabular data faces a myriad of challenges. Challenges that were previously identified in tabular domain are summarized below.

Mixed Attribute Type. Real data contains a mix of continuous, categorical, and discrete attributes [1]. One of the main challenges of tabular data is handling the mixed data types for data generation. Methodologies applied towards continuous attributes may not always work well with other data types, such as nominal types. A robust framework that incorporates the changing data type is needed.

Data Fidelity. Data fidelity plays an important role when synthetic samples are generated, especially in electronic health records (EHR) [2]. For example, synthetic data should not have rows with prostate cancer diagnoses for women. It is one of the most challenging criteria to tackle in synthetic data generation. The data generation framework should achieve data fidelity to ensure the accuracy of the artificial samples.

Domain-specific. Tabular data is domain-specific. Attributes and their values depend on the domain, such as EHR data. Data curation based on the domain's accepted data range is a daunting and costly task. On the other hand, the Curated data requires some validation from any domain expert. The data generation framework should generate data without any user intervention irrespective of the domain. Synthetic data should be in the allowed data range with high data fidelity.

Task-specific. The data generation framework should not be task-specific. Most of the frameworks generate data for either classification or regression tasks using the target variables. In

* Corresponding author.
E-mail addresses: jsivaku1@binghamton.edu (J. Sivakumar), r.karthik@vit.ac.in (K. Ramamurthy), menaka.r@vit.ac.in (M. Radhakrishnan), dhwon@binghamton.edu (D. Won).

https://doi.org/10.1016/j.knosys.2023.110956
0950-7051/© 2023 Elsevier B.V. All rights reserved.

general, the framework should generate synthetic data for any data-driven modeling task in an unsupervised manner.

Mode Collapse. Tabular data faces the problem of imbalanced categorical attributes. When synthetic samples are generated for such imbalanced attributes, only majority classes are generated, which frequently leads to mode collapse [1,3]. The minority classes may not be encountered as often as the majority classes, resulting to poor synthetic data quality. The data generation framework must generate minority classes as frequently as the majority classes by picking up different modes in the data to avoid mode collapse.

Statistical Similarity. The statistical similarity of synthetic data should be close to the actual data distributions [4]. A quality synthetic data will be similar and realistic proxy for real data. Therefore, the data can be safely used for any research and analysis. The closeness of the synthetic data is measured statistically and through machine learning utility.

Data privacy. Data privacy protects sensitive information from any adversarial attacks [4,5]. Healthcare records face re-identification attacks when one or more sensitive attributes are disclosed. Quality synthetic data retains data privacy without disclosing sensitive information while achieving data fidelity. The generated synthetic data generated should ideally achieve high privacy and high fidelity.

Deep Learning for Small Data. Deep learning-based synthetic data generation frameworks for tabular data are limited. In addition, using small datasets for training the generative models is another challenge. A huge number of samples are required for training deep learning models. A deep learning-based method that uses small datasets as an input for synthetic data generation was not explored previously [6].

We propose a generative model, GenerativeMTD, to tackle these challenges. We extend our previous algorithm, k-nearest neighbor mega-trend diffusion (kNNMTD) [7]. A synthetic data generation methodology for small datasets that uses k-nearest neighbor (kNN) algorithm and mega-trend diffusion (MTD). GenerativeMTD uses pseudo-real data that is uniformly generated based on the domain ranges estimated from the real data. Instead if using the real data, pseudo-real data is used for model training to avoid privacy violation. The way the pseudo-real data is generated incorporates the semantics of the real data. A sufficient number of samples that overcome small dataset problem for model training can be generated. GenerativeMTD translates the pseudo-real data towards real data through optimization of the proposed reconstruction and divergence loss. GenerativeMTD uses a variational autoencoder-based generative adversarial network architecture (VAE-GAN). An advantage of VAE-GAN is to use the latent representations from the real data through the encoder. The autoencoder part acts as a generator and the critic network discerns if the data is either real or fake. Both of these networks are trained simultaneously similar to GAN. The framework is the first-of-a-kind approach that provides a straightforward way to handle small datasets using deep learning. In addition, a framework to evaluate synthetic data through different data quality metrics is showcased. The contributions of our research are as follows:

- GenerativeMTD is a first-of-a-kind deep learning framework that uses small datasets as input for synthetic data generation.
- GenerativeMTD uses pseudo-real data to train the deep learning model instead of real data, which enhances the privacy of synthetic data.
- GenerativeMTD consistently generates statistically similar data with high distance-based privacy compared to the other benchmark models that exist for tabular data.
- The framework is able to generate synthetic data irrespective of the domain and task with no user intervention.
- GenerativeMTD handles mixed attributes and generate minority classes to avoid mode collapse.

In the next section we introduce the existing synthetic data generation methodologies for tabular datasets. In the methodology section, the algorithmic steps of pseudo real data generation is discussed followed by the technical details of the generator, critic, and the network architecture. We discuss the different evaluation metrics of the synthetic data that are used in this paper. In the experimental setup, different datasets to evaluate the proposed method are showcased. The results section contains the detailed description of how the final synthetic data is chosen for each of these datasets. We conclude with some future directions to improve GenerativeMTD.

2. Related work

Different synthetic data generation techniques that emerged in the last two decades are fuzzy theory-based as well as machine learning-based. Some data generation frameworks that are tool-based (such as R programming) are developed to synthesize multi-label datasets and noise simulation of chemical data [8–10]. But generative models are in the spotlight due to the quality of the synthetic data generated. Variational autoencoders are one of the generative models with two networks: *Encoder* and *Decoder*. Encoder takes the real data as input. The network then maps the input data into latent space $Z \sim \mathcal{N}(0, \mathbf{I}), Z \in \mathbb{R}^q$, where q is some arbitrary dimension. Decoder generates reconstructed sample $\tilde{X} \in \mathbb{R}^p$ from the latent representation, where p is the real-data dimension.

$$\begin{aligned} Z &\sim Enc(X) = Q(Z|X) \\ \tilde{X} &\sim Dec(Z) = P(X|Z) \end{aligned} \tag{1}$$

VAE tries to maximize the evidence lower bound (ELBO), as given by

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{Q(Z|X)}[\log P(X|Z)] - D_{KL}(Q(Z|X) \,\|\, P(Z)) \tag{2}$$

The first term corresponds to reconstruction error. In the second term, Posterior distribution $Q(Z|X)$ and latent distribution $P(Z)$ approach each other. A variety of VAE-based methods have been proposed in the recent times. TVAE [1] is the traditional VAE network applied on the tabular data using ELBO objective. Info-VAE [11] alters the traditional VAE by introducing mutual information between X and Z. The new training objective is given by

$$\begin{aligned} \mathcal{L}_{\text{InfoVAE}} = -\lambda \cdot D_{KL}(Q(Z|X) \,\|\, P(Z)) - \mathbb{E}_{Q(Z|X)}[\log P(X|Z)] \\ + \alpha \cdot I_Q(X; Z) \end{aligned} \tag{3}$$

where $I_Q(X;Z)$ is the mutual information between X and Z under the distribution $Q(X,Z)$.

Autoencoders also use optimal transport (OT) by replacing the traditional reconstruction error [12]. The OT-cost divergence $W_c(P_X, P_{G(Z)})$ is computed through p-Wasserstein distances. Wasserstein Autoencoder (WAE) is one such generative method that incorporates OT-cost divergence [13,14]. The objective function of WAE is given by

$$\min_{G} \min_{Q(Z|X)} \mathbb{E}_{X \sim P_r} \mathbb{E}_{Z \sim Q(Z|X)}[c(X, G(Z))] + \lambda \cdot \mathcal{D}_Z(P_Z, Q_Z) \tag{4}$$

where $c(X, G(Z))$ is any measurable non-negative cost, Q is any non-parametric set of probabilistic encoders, $\mathcal{D}_Z$ is an arbitrary divergence between Q_Z and P_Z, $\lambda > 0$ is a Lagrange multiplier. The choice of divergence $\mathcal{D}_Z(P_Z, Q_Z)$ can be maximum mean discrepancy (MMD) [15–17] or a discriminator-trained adversarial loss in the latent space instead of using KL-divergence [13].

Alternatively, generative adversarial networks (GAN) generate synthetic data [18]. GANs have two networks: *Generator* and *Discriminator*. Generator G generates data from the latent representation $Z \sim \mathcal{N}(0, \mathbf{I}), Z \in \mathbb{R}^q$ that maps into the data space $\tilde{X} \in \mathbb{R}^p$, where q is some arbitrary dimension and p is the real data dimension. The discriminator D discerns if the data is either generated or fake. Both the networks play a zero-sum minimax game with the payoff function $V(G, D)$

$$\min_G \max_D V(G, D) = \mathbb{E}_{X \sim \mathcal{P}_r}[\log D(X)] + \mathbb{E}_{Z \sim \mathcal{P}_Z}[\log(1 - D(G(Z)))] \tag{5}$$

The goal of the generator is to generate synthetic data similar to real data. The goal of the discriminator is to correctly distinguish the fake samples from the real ones. At convergence, the generator's samples $\tilde{X}$ are indistinguishable from the real data, and the discriminator merely guesses if the sample is real or fake. Some GAN models have been recently introduced that generate tabular data, despite fake images. medGAN [19] and corrGAN [20] generate discrete patient data. ehrGAN [21] generates augmented patient data. PATE-GAN [5] generates differentially private synthetic data. Additionally, privacy-preserving GAN (pGAN) is proposed to overcome the privacy constraints on EHR data [22]. VEEGAN [23], TableGAN [4], CTGAN [1], and CTAB-GAN [24] generate synthetic data for tabular datasets.

VEEGAN, TableGAN, TVAE, and CTGAN are the benchmark models in this research. VEEGAN is the GAN-based architecture that avoids mode collapse, which is an important contribution in tabular domain. In addition to a generator and a discriminator, a reconstructor network is used to map the true data and generated data to a Gaussian distribution. TableGAN has three different networks: generator, discriminator, and classifier. The purpose of generator and discriminator are the same, whereas the classifier is trained to learn the correlations between the labels and other attributes. It teaches the generator if the record is semantically correct. Because discriminator does not learn the semantic integrity, the classifier effectively corrects the generator to impose the semantic integrity. CTGAN and TVAE are an algorithmic methodology proposed under GAN and VAE architectures, respectively. CTGAN take conditional vectors as an input for the categorical columns. The random sampling during the training is adjusted to conditionally sample around the minority classes. This is one of the challenges when the tabular data has mixed attribute types with high class imbalance. The synthetic data should be generated for different modes of the real data. In TVAE, the joint distribution of the conditional vectors is output in the encoder part. The decoder then generates the synthetic data.

GANs achieve differential privacy to prevent adversary attacks [25]. Classification enhancement GAN (CEGAN) generates synthetic data for minority classes and improves the prediction accuracy for imbalanced data conditions [26]. Although GANs are efficient in the generation of synthetic data, the discriminator overfits to small datasets, which makes its feedback meaningless for the generator [27].

Apart from VAEs and GANs, a deep generative model similar to GAN-like training with VAE architecture was proposed previously. VAE-GAN [28] is a variational autoencoder network that is coupled with GAN. This generative model trains VAE network (generator) and discriminator simultaneously. The real data is encoded through the encoder network. The encoded $Z \sim \mathcal{N}(0, \mathbf{I})$ is the input to the decoder. The decoded data is the fake data. The discriminator discerns if the sample is fake or real. An advantage of VAE-GAN is to use the discriminator to improve the VAE reconstruction objective. The VAE part offers the capability of using the real data to improve the statistical similarity of the data. The objective function for VAE-GAN becomes

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{GAN}}, \\ \mathcal{L}_{\text{prior}} &= \mathcal{D}_{KL}(Q(Z|X) \,\|\, P(Z)) \\ \mathcal{L}_{\text{recon}} &= -\mathbb{E}_{Q(Z|X)}[\log P(D_l(X|Z))] \\ \mathcal{L}_{\text{GAN}} &= \log(D(X)) + \log(1 - D(G(Z))) \end{aligned} \tag{6}$$

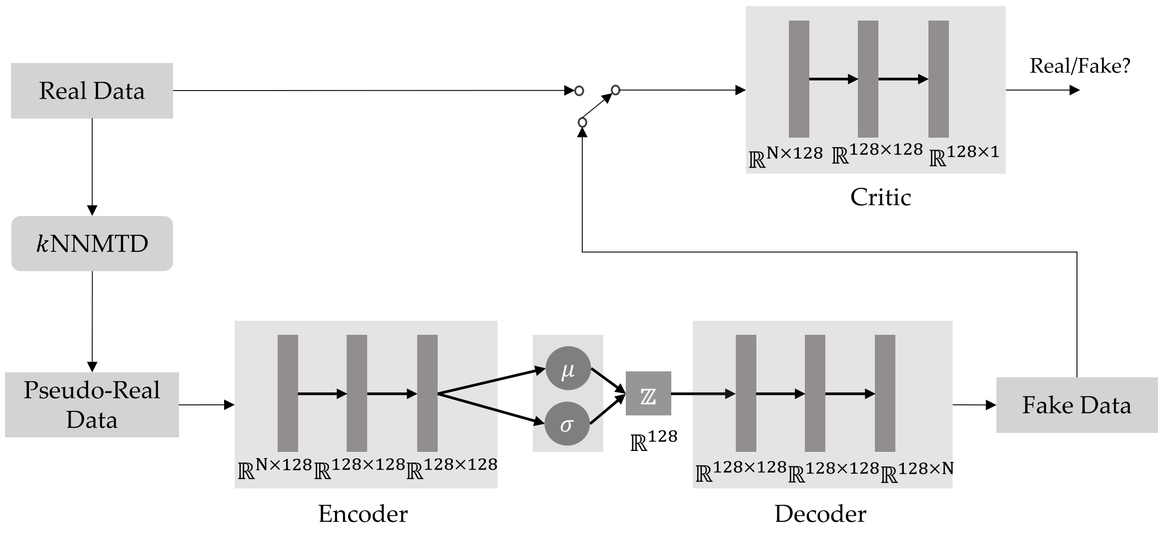
D_{KL} is KL-divergence, $D_l(X)$ represents lth layer hidden representation of the discriminator.

Tabular synthetic data is evaluated in different aspects to assess the quality. Pairwise correlation difference (PCD) measures the statistical similarity of tabular synthetic data and real data [2]. Goncalves et al. [2] summarized other metrics to evaluate synthetic data such as Kullback–Leibler (KL) divergence, log-cluster metric, support coverage, and cross-classification. In addition, Kolmogorov–Smirnov test (KSTest) and Chi-Squared test (CSTest) [29] are used to compare the distributions of the real and synthetic column. For continuous columns, the empirical cumulative distribution function (CDF) is used to compare the real and synthetic data distributions. Machine learning utility is computed through training on synthetic and test on real (TSTR) [30], training on real and test on synthetic (TRTS), training on real and test on real (TRTR), and training on synthetic and test on synthetic (TSTS) [31]. When the machine learning utility is higher, the quality of synthetic data is high as well. Privacy of synthetic data is measured through distance-based metrics or through differential privacy. Because the scope of the paper is distance-based, the distance to the closest record (DCR) [4] and nearest neighbor distance ratio (NNDR) [24,32,33] are the two distance-based privacy measures used in this research. Evaluation of tabular synthetic data is an open-ended research topic. In this research, a data generation framework that automatically incorporates some evaluation metrics is proposed.

All these previously proposed methods come with a variety of limitations. The privacy and machine learning utility of TVAE is not well addressed. VEEGAN and TableGAN suffer from mode collapse. GAN-based architectures are not reliable to train small datasets. This is the foremost issue in the existing tabular data generation methods. In addition, statistical similarity and privacy are not the main objectives for these methods. The use of real data for training VAEs has a risk of privacy violations. A known property of VAE is that the optimal decoder will memorize training data in the limit of infinite capacity [34,35] as will a deterministic autoencoder [36]. No real data (but pseudo-real) is used for training and generation of synthetic data in the proposed method. As for the case of small datasets, synthetic data generation that uses deep learning models is infeasible due to the sample size. It will be tedious to capture all the non-linear relationships in the data. This drawback is mitigated through GenerativeMTD.

3. Methodology

GenerativeMTD is proposed to be one of the generative models that uses VAE-GAN-like architecture as shown in Fig. 1. GenerativeMTD first generates pseudo-real data that will act as a training set. The encoder takes the pseudo-real data, X' as an input. The pseudo-real data is mapped into the latent space $Z \sim \mathcal{N}(0, \mathbf{I}), Z \in \mathbb{R}^q$, where q is some arbitrary dimension. The decoder maps the latent variables into the data space, $\tilde{X} \in \mathbb{R}^p$, where p is the real data dimension. In this methodology, critic is used instead of a discriminator [37]. Critic provides a more informative measure of distance between real and synthetic data distributions. Critic outputs a continuous value rather than a binary value which represents a degree of realism of the data. This allows for a more stable and reliable training process. Another advantage of using



Real Data
Real/Fake?
$\mathbb{R}^{N\times128}$ $\mathbb{R}^{128\times128}$ $\mathbb{R}^{128\times1}$
Critic
kNNMTD
Pseudo-Real Data
μ
σ
$\mathbb{Z}$
$\mathbb{R}^{128}$
Fake Data
$\mathbb{R}^{N\times128}$ $\mathbb{R}^{128\times128}$ $\mathbb{R}^{128\times128}$
Encoder
$\mathbb{R}^{128\times128}$ $\mathbb{R}^{128\times128}$ $\mathbb{R}^{128\times N}$
Decoder

Fig. 1. GenerativeMTD.

a critic is that it is easier to train than a discriminator. This is because the critic is not required to output a probability value, which can be difficult to optimize. The critic network outputs an unbounded score. Henceforth, the VAE part of GenerativeMTD will be referred to as generator and critic is referred to as it is.

3.1. Generating pseudo-real data

The first and foremost step of GenerativeMTD is to generate pseudo-real data. Pseudo-real data replaces real data for training purposes. When the dataset size is small, training using deep learning techniques becomes infeasible. To avoid using the real data for training directly and enhance the dataset size, pseudo-real data is generated as an intermediate synthetic data. The objective of GenerativeMTD is to translate the pseudo-real data into synthetic data that is as realistic as real data. Pseudo-real data is generated through k-nearest neighbor mega-trend diffusion (kNNMTD) [7]. We introduced kNNMTD, a synthetic generation framework for small datasets, in our previous work. The steps of kNNMTD are slightly modified in this paper. The modification is made to accommodate the required number of pseudo samples for model training. kNNMTD's pseudo-real data retains the semantics of the real data. The generated samples are around the vicinity of the real data points.

Consider a dataset $\mathbf{x} \in \mathbb{R}^{n\times p}$. For some k, k neighbors are identified for each data point $x_{i,j}$. The upper and lower bound domain ranges from the neighboring subsamples for each $x_{i,j}$ are identified as follows:

$$\begin{aligned} a_{(i,j)} &= u_{set}^{(i,j)} - Skew_L^{(i,j)} \times \sqrt{-2 \times \hat{s}_x^2/N_L^{(i,j)} \times \ln(10^{-20})} \\ b_{(i,j)} &= u_{set}^{(i,j)} + Skew_U^{(i,j)} \times \sqrt{-2 \times \hat{s}_x^2/N_U^{(i,j)} \times \ln(10^{-20})} \\ &1 < N_L^{(i,j)} < \infty,\ 1 < N_U^{(i,j)} < \infty, \\ \text{where } \hat{s}_x^2 &= \sum_{i=1}^{k}(x_i - \bar{x}_k)^2/k - 1, \\ u_{set}^{(i,j)} &= (min_{(i,j)} + max_{(i,j)})/2, \\ Skew_L^{(i,j)} &= N_L^{(i,j)}/(N_L^{(i,j)} + N_U^{(i,j)}), \\ Skew_U^{(i,j)} &= N_U^{(i,j)}/(N_L^{(i,j)} + N_U^{(i,j)}) \end{aligned} \tag{7}$$

where sub or superscript (i,j) represents the parameters of the neighbors of $x_{i,j}$, $N_L^{(i,j)}$ represent the number of data points in the sample set that are less than $u_{set}^{(i,j)}$, and $N_U^{(i,j)}$ represent the number of data points in the sample set that are greater than $u_{set}^{(i,j)}$. In the case of $\hat{s}_x^2 = 0$, $a_{(i,j)}$ and $b_{(i,j)}$ is given by

$$\begin{aligned} a_{(i,j)} &= min_{(i,j)}/5 \\ b_{(i,j)} &= max_{(i,j)} \times 5 \end{aligned} \tag{8}$$

where $min_{(i,j)}$ and $max_{(i,j)}$ are the minimum and maximum values of the neighboring subsamples of (i,j)th instance. When the values of a and b computed here do not cover the original subsample's domain ranges, the lower bound and upper bound with respect to the subsamples are computed using Eq. (9). The estimated lower bound (LB) and upper bound (UB) become

$$\begin{aligned} LB_{(i,j)} &= \begin{cases} a_{(i,j)} & \text{if } a_{(i,j)} \le min_{(i,j)}, \\ min_{(i,j)} & \text{if } a_{(i,j)} > min_{(i,j)} \end{cases} \\ UB_{(i,j)} &= \begin{cases} b_{(i,j)} & \text{if } b_{(i,j)} \ge max_{(i,j)}, \\ max_{(i,j)} & \text{if } b_{(i,j)} < max_{(i,j)} \end{cases} \end{aligned} \tag{9}$$

The membership function (MF) is given by,

$$MF(x'_{(i,j)}) = \begin{cases} \frac{x'_{(i,j)} - LB_{(i,j)}}{u_{set}^{(i,j)} - LB_{(i,j)}} & \text{if } x'_{(i,j)} \le u_{set}^{(i,j)}, \\ \frac{UB_{(i,j)} - x'_{(i,j)}}{UB_{(i,j)} - u_{set}^{(i,j)}} & \text{if } x'_{(i,j)} > u_{set}^{(i,j)}, \\ 0 & \text{if } x'_{(i,j)} < LB_{(i,j)} \text{ or} \\ & x'_{(i,j)} > UB_{(i,j)} \end{cases} \tag{10}$$

Using the membership function from Eq. (10), the samples are generated using plausibility assessment mechanism (PAM). The samples x' are generated using $\mathcal{U}(LB_{(i,j)}, UB_{(i,j)})$ for a given data point $x_{i,j}$ according to Eq. (9). The MF value for x' is computed using Eq. (10). Generate a uniformly distributed random value $(rs) \sim \mathcal{U}(0,1)$. If $MF(x') > rs$, then the sample x' is retained; otherwise, the sample is discarded. This step in GenerativeMTD ensures that the samples retain the closeness of the real data distribution.

In our previous work, kNNMTD offers a final step that is highly very imperative for statistical similarity. The k-nearest neighbors from x' with respect to the original sample $x_{i,j}$ are selected as the final accepted artificial samples. By doing so, we further push the semantics of the synthetic data towards real data. But this very step is ignored in GenerativeMTD. In GenerativeMTD, synthetic data generated is translated by optimizing the loss functions (in Section 3.2). The translation procedure optimizes the final synthetic data on statistical similarity and privacy. GenerativeMTD does not choose any nearest neighbors from the retained samples

x'. This step ensures that the final dataset does not contain any copy of the real data. All the generated samples x' that pass $MF(x') > rs$ are used towards building the pseudo-real data.

The above procedure starts from the first row $i = 1$ and is iterated to each column $j = 1$ to p. Then the algorithm moves to the second row and so forth until the whole data $x_{n,p}$ is synthesized sequentially. Therefore, the retained x' samples from the above procedure are first concatenated column-wise. This column-wise concatenated data is stacked row-wise after each row is synthesized. The final pseudo data is the concatenated data at $x_{n,p}$.

The size of the pseudo-real (training) data X' is increased through this procedure. The pseudo-real data is large enough to train the deep learning model to output the final synthetic data after training. This pseudo-real data is fed into the encoder. Simultaneous training of the autoencoder and critic translates the pseudo-real data closer to the real dataset. Because the procedure does not use the real dataset for training, the privacy of the real data should be retained.

3.2. Generator

The pseudo-real samples X' generated from the above-mentioned procedure are the input to the generator. Structural properties of pseudo-real data are integrated through *k*NNMTD [7]. The pseudo-real data is mapped to latent representations for quality synthetic data through the encoder. The model representation for the generator part is given by

$$Z' \sim Enc(X') = Q(Z'|X')$$
$$\tilde{X} \sim Dec(Z') = P(X|Z')$$

The real data is mapped onto the latent space only to compute the divergence error. Similarly, the latent space representation for the real dataset X is given by

$$Z \sim Enc(X) = Q(Z|X)$$

VAE approximates the posterior distribution closer to some prior by minimizing KL-divergence. But KL-divergence is unstable with respect to deformations of distributions' support [38]. Sinkhorn algorithm [39] provides a way to account for the distances of the distribution in the feature space. We introduce the usage of Sinkhorn iterations [40] to compute divergent loss and show the effectiveness of the proposed generative model experimentally. In our proposed method, Sinkhorn divergence to compute divergence loss using real and pseudo-real data is illustrated in the following section.

3.2.1. Divergence loss

Consider the random variables X, Y, Z and their corresponding probability distributions P_X, P_Y, P_Z from the metric spaces $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$. We want to map the real data and pseudo-real data onto the latent space $Z = Q(Z|X)$ and $Z' = Q(Z'|X')$. P_Z denotes the prior of the real data. $Q_{Z'} = Q(Z'|X')P_{X'} = \mathbb{E}_{X'\sim P_{X'}} Q(Z'|X')$ denote the approximated posterior distributions of the pseudo-real data over the metric space $\mathcal{Z}$. The optimal transport (OT) metric between the two distributions P_Z and $Q_{Z'}$ is modified as follows,

$$S_c(P_Z, Q_{Z'}) = \inf_{\Gamma\in\prod(P_Z,Q_{Z'})} \mathbb{E}_{(Z,Z')\sim\Gamma}[c(Z, Z')] \tag{11}$$

where c is the p-Wasserstein distance for $p \geq 1$, Γ is the coupling distribution of the marginals, and $\prod(P_Z, Q_{Z'})$ is the set of joint distributions between the marginals. The regularized OT problem is proposed in Genevay et al. [40], which is given by

$$S_{c,\varepsilon}(P_Z, Q_{Z'}) = \inf_{\Gamma\in\prod(P_Z,Q_{Z'})} \mathbb{E}_{(Z,Z')\sim\Gamma}[c(Z, Z')] + \varepsilon \cdot \mathrm{KL}(\Gamma, P_Z \otimes Q_{Z'}) \tag{12}$$

where ε is the regularization parameter. After removing the entropic bias, we arrive at Sinkhorn divergence [40]

$$S_{c,\varepsilon}(P_Z, Q_{Z'}) = 2S_{c,\varepsilon}(P_Z, Q_{Z'}) - \left(S_{c,\varepsilon}(P_Z, P_Z) - S_{c,\varepsilon}(Q_{Z'}, Q_{Z'}) \right) \tag{13}$$

The following is the behavior with respect to ε:

$$\begin{aligned} \bar{S}_{c,\varepsilon}(P_Z, Q_{Z'}) &\xrightarrow{\varepsilon\to 0} 2S_c(P_Z, Q_{Z'}), \\ \bar{S}_{c,\varepsilon}(P_Z, Q_{Z'}) &\xrightarrow{\varepsilon\to\infty} \mathrm{MMD}_{-c}(P_Z, Q_{Z'}) \end{aligned} \tag{14}$$

$S_{c,\varepsilon}$ interpolates between the optimal transport divergence and maximum mean discrepancy (MMD). MMD_{-c} is the MMD distance cost kernel from the optimal transport problem. The approximation of Sinkhorn divergence $S_{c,\varepsilon}$ can be estimated through Sinkhorn algorithm [39]. Through mini-batch sampling of size M, $\hat{P}_Z = \frac{1}{M}\sum_{i=m}^{M} \delta_{Z_m}$, $\hat{Q}_{Z'} = \frac{1}{M}\sum_{m=1}^{M} \delta_{\tilde{Z}_m}$. The optimal regularized OT cost $\bar{S}_{c,\varepsilon}(P_Z, Q_{Z'})$ with $\varepsilon \geq 0$ is given as

$$\mathcal{L}_{\text{sink}} = R^* := \operatorname*{arg\,min}_{R\in S_M} \frac{1}{M}\langle R, \tilde{C}\rangle_F - \varepsilon \cdot H(R) \tag{15}$$

where $\tilde{C} = \tilde{c}(\tilde{Z}_i, Z_j)$ is the cost matrix. In our proposed method, we use $\tilde{c}(\tilde{Z}_i, Z_j) = \left\|\tilde{Z}_i - Z_j\right\|_2^2$. R is a doubly stochastic matrix defined as $S_M = \{R \in \mathbb{R}_+^{M\times M} \mid R\mathbb{1} = \mathbb{1}, R^T\mathbb{1} = \mathbb{1}\}$, $\langle\cdot,\cdot\rangle$ denotes a Frobenius inner product, and $\mathbb{1}$ is vector of ones. $H(R) = -\sum_{i,j=1}^{M} R_{i,j}\log R_{i,j}$ is the entropy of R. An advantage of regularized OT is that it becomes efficiently solvable using Sinkhorn algorithm. Thus, it leads to a differentiable loss function. Sinkhorn iteration [40,41] provides differential capabilities that can be utilized in the back propagation of the generator training. The divergence loss $\mathcal{L}_{\text{sink}}$ for the generator is computed using the Sinkhorn iteration. Because ε-regularized optimal transport problem is a strong convex function due to entropy, Sinkhorn algorithm [42] returns a unique optimal solution to Eq. (15). As $\varepsilon \to 0$ and $L \to +\infty$ Sinkhorn iteration approaches a solution for Eq. (15) with linear convergence rate. Sinkhorn algorithm is a fixed-point algorithm that can be implemented through matrix multiplications (See Algorithm 2). The unique solution for Eq. (15) is

$$R^*_{i,j} = u_i \cdot K_{i,j} \cdot v_j \tag{16}$$

where K is the Gibbs kernel. The final loss after L iterations is then given by

$$\begin{aligned} R^*_{i,j} &= \mathrm{diag}(u) \cdot K \cdot \mathrm{diag}(v) \\ K_{i,j} &= e^{\tilde{C}_{i,j}/\epsilon} \end{aligned} \tag{17}$$

Starting with $v^{(0)} = \mathbb{1}_m$, $l \leftarrow 0$. For every iteration,

$$\begin{aligned} u_{l+1} &= \frac{\mathbb{1}_m}{K v_l} \\ v_{l+1} &= \frac{\mathbb{1}_m}{K^T u_{l+1}} \end{aligned} \tag{18}$$

3.2.2. Reconstruction loss

The expected log-likelihood loss in the baseline VAE is replaced with maximum mean discrepancy [15–17]. As opposed to using MMD as a divergence loss, a first-of-a-kind usage of MMD for reconstruction loss is proposed in this research. MMD matches all the orders of statistics between real and generated distributions. This makes the generated data indistinguishable.

MMD compares the two distributions by comparing their moments. MMD is applied only to the continuous column types. The loss is given by

$$\mathcal{L}_{\text{MMD}} = \text{MMD}_k(X, \tilde{X}) = \mathbb{E}_{X\sim P_X}[k(X, \tilde{X})] + \mathbb{E}_{\tilde{X}\sim Q_{\tilde{X}}}[k(\tilde{X}, \tilde{X})] - 2\mathbb{E}_{X\sim P_X, \tilde{X}\sim Q_{\tilde{X}}}[k(X, \tilde{X})] \tag{19}$$

where X is the real data, $\tilde{X}$ is the final fake data generated from GenerativeMTD, and $k(\cdot,\cdot)$ is any positive definite kernel. Typically, $k(\cdot,\cdot)$ is assumed to be a Gaussian kernel $k(\cdot,\cdot) = \exp\left(-\frac{1}{2\sigma}\|X - \tilde{X}\|^2\right)$ with bandwidth parameter σ. $\mathcal{L}_{\text{MMD}} = 0$ if and only if $P_X = Q_X$.

Cross-entropy loss is applied to the one-hot encoded categorical columns in the tabular data. For C categorical levels in an attribute, the loss is computed for x and $\tilde{x}$. The cross-entropy loss is

$$\mathcal{L}_{\text{entropy}} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{C} x_{i,j}\log(\tilde{x}_{i,j}) \tag{20}$$

where M is mini-batch size, C is the number of classes for an attribute, and $x_{(i,j)}$ and $\tilde{x}_{(i,j)}$ represent the value in jth column of a one-hot encoded data. The final loss for the generator in GenerativeMTD is given by

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{sink}}}_{\text{Divergence loss}} + \underbrace{\mathcal{L}_{\text{MMD}} + \mathcal{L}_{\text{entropy}}}_{\text{Reconstruction loss}} \tag{21}$$

Algorithm 1 provides the GenerativeMTD steps. The Sinkhorn iteration algorithm is adopted from Genevay et al. [40], which is given in Algorithm 2.

3.3. Critic

Instead of a discriminator, critic is incorporated [37]. The network is trained to minimize WGAN loss with gradient penalty, which is given by

$$\mathcal{L} = \underbrace{\mathbb{E}_{\tilde{X}\sim\mathbb{P}_g}[D(\tilde{X})] - \mathbb{E}_{X\sim\mathbb{P}_r}[D(X)]}_{\text{Critic loss}} + \underbrace{\lambda\cdot\mathbb{E}_{\tilde{X}\sim\mathbb{P}_g}[(\|\nabla_{\tilde{X}}D(\tilde{X})\|_2 - 1)^2]}_{\text{Gradient penalty}} \tag{22}$$

The first term is the critic loss. The second term (gradient penalty) is added to the loss to prevent vanishing or exploding gradient problems. The vanishing gradient problem occurs when the gradient is zero and the network stops learning. An Exploding gradient occurs when the gradient penalty exponentially increases when it is backpropagated. When the weights of the critic network are no longer updated due to vanishing or exploding gradients, the generator will stop the learning process and the training may not converge. Ideally, a converging critic network will provide useful feedback to the generator's training. By letting the norm of the gradient penalty term approach 1, critic training will be improved without leading the training to saturation.

3.4. Network structure

For a tabular data with set of $D_1, D_2, \ldots, D_{N_d}$ discrete attributes and $C_1, C_2, \ldots, C_{N_c}$ continuous attributes, $N = N_c + N_d$, where N_c is the number of continuous columns, N_d is the number of discrete columns. When a discrete column $(1 \leq i \leq N_d)$ is one-hot encoded, the discrete attributes in the tabular data for $1 \leq j \leq k$ levels become

$$D_{1,1} \oplus \cdots \oplus D_{1,k} \oplus \cdots \oplus D_{i,1} \oplus \cdots \oplus D_{i,k} \oplus \cdots \oplus D_{N_d,1} \oplus \cdots \oplus D_{N_d,k}$$

Algorithm 1: GenerativeMTD with $M = n, \delta = 10, \gamma = 0.0002, \mu = 0.9, \lambda = 10^{-6}, n_{critic} = 1, n_{obs} = 100$, epoch = 200

```
Input: Encoder weights $e_0$, Decoder weights $d_0$, Critic
       weights $c_0$, Real data $\{x_i\}_{i=1}^n$, Batch Size $M$,
       Regularization parameter $\varepsilon$, Sinkhorn Iteration $L$,
       Gradient penalty coefficient $\delta$, Learning rate $\gamma$,
       Momentum $\mu$, Weight decay $\lambda$, Number of critic
       iterations $n_{critic}$, Number of observation for each
       instance $n_{obs}$, epoch
Output: $\tilde{\mathbf{x}}, e, d$
$e \leftarrow e_0$
$d \leftarrow d_0$
$c \leftarrow c_0$
// Generate pseudo-real samples
$\mathbf{x}' \leftarrow k\mathtt{NNMTD}(x_i, n_{obs})$
$N \leftarrow n_{obs} * n * 10$
while d not converged or epoch not done do
    for $i = 1, 2 \ldots, N/M$ do
        Sample $\{\mathbf{x}'_i\}_{i=1}^M \sim P_{\mathbf{x}'}$
        for $t = 1, 2, \ldots, n_{critic}$ do
            $\mathbf{z}_{fake} \leftarrow Enc(\mathbf{x}'_i)$
            $\tilde{\mathbf{x}} \leftarrow Dec(\mathbf{z}_{fake})$
            $\mathcal{L}_{critic} \leftarrow D_c(\tilde{\mathbf{x}}) - D_c(\tilde{\mathbf{x}}) + \delta\cdot(\left\|\nabla_{\tilde{\mathbf{x}}}D_c(\tilde{\mathbf{x}})\right\|_2 - 1)^2$
            $c_{t+1} \leftarrow c_t - \gamma\cdot\mathrm{SGD}(\nabla_c\mathcal{L}_{critic}, \mu, \lambda)$
        $\mathbf{z}_{real} \leftarrow Enc(\mathbf{x})$
        $\mathbf{z}_{fake} \leftarrow Enc(\mathbf{x}'_i)$
        $\tilde{\mathbf{x}} \leftarrow Dec(\mathbf{z}_{fake})$
        // Algorithm 2
        $\mathcal{L}_{sink} \leftarrow \tilde{S}_{\tilde{c},\varepsilon}(\mathbf{z}_{fake}, \mathbf{z}_{real})$
        // Eq. (19)
        $\mathcal{L}_{MMD} \leftarrow \mathrm{MMD}_k(\mathbf{x}, \tilde{\mathbf{x}})$
        // Eq. (20)
        $\mathcal{L}_{entropy} \leftarrow \frac{1}{M}\sum_{i=1}^M \mathtt{CrossEntropy}(\mathbf{x}, \tilde{\mathbf{x}})$
        // Eq. (21)
        $\mathcal{L} \leftarrow \mathcal{L}_{sink} + \mathcal{L}_{MMD} + \mathcal{L}_{entropy} - \frac{1}{M}\sum_{i=1}^M D_c(\tilde{\mathbf{x}}_i)$
        $(e, d) \leftarrow (e, d) - \gamma\cdot\mathrm{SGD}(\nabla_{(e,d)}\mathcal{L}, \mu, \lambda)$
```

Algorithm 2: Sinkhorn Iteration with $\varepsilon = 0.01, L = 100$

```
Input: $\{\tilde{z}_i\}_{i=1}^M \sim \mu_e$, $\{z_j\}_{j=1}^M \sim \nu, \varepsilon, L$
Output: $R^*, \tilde{C}$
$\tilde{C}_{i,j} = \tilde{c}(\tilde{z}_i, z_j)\ \forall i, j$
$K = \exp(-\tilde{C}/\varepsilon)$, $u \leftarrow 1$
for $l = 1, 2, \ldots, L$ do
    $v \leftarrow \mathbf{1}/(K^T u)$
    $u \leftarrow \mathbf{1}/(Kv)$
$R^* \leftarrow \mathrm{Diag}(u)\cdot K\cdot\mathrm{Diag}(v)$
```

The network structure for encoder is given by

$$\begin{cases} h_0 = \mathbf{x}' \\ h_1 = \mathtt{Dropout}_{0.5}(\mathtt{Leaky}_{0.1}(\mathtt{FC}_{N\to128}(h_0))) \\ h_2 = \mathtt{Dropout}_{0.5}(\mathtt{Leaky}_{0.1}(\mathtt{FC}_{128\to128}(h_1))) \\ h_3 = \mathtt{Dropout}_{0.5}(\mathtt{Leaky}_{0.1}(\mathtt{FC}_{128\to128}(h_2))) \\ \mu = FC_{128\to128}(h_3) \\ \sigma = \exp(\frac{1}{2}(FC_{128\to128}(h_3))) \\ \mathbf{z}' \sim \mathcal{N}(\mu, \sigma\mathbf{I}) \end{cases}$$

The decoder network is given by

$$\begin{cases} h_1 = \texttt{Dropout}_{0.5}(\texttt{Leaky}_{0.1}(\texttt{FC}_{128\to128}(\mathbf{z}'))) \\ h_2 = \texttt{Dropout}_{0.5}(\texttt{Leaky}_{0.1}(\texttt{FC}_{128\to128}(h_1))) \\ h_3 = \texttt{Dropout}_{0.5}(\texttt{Leaky}_{0.1}(\texttt{FC}_{128\to N}(h_2))) \\ \tilde{\mathbf{x}} = \texttt{activation}(h_3) \end{cases}$$

The activation functions are applied as follows

$$\texttt{activation}(h_{3_i}) = \begin{cases} \texttt{gumbel}_{0.2}(h_{3_i}), & \text{if } 1 \le i \le N_d \\ \texttt{tanh}(h_{3_i}), & \text{if } 1 \le i \le N_c \end{cases}$$

The network structure for critic is given by

$$\begin{cases} h_0 = \tilde{\mathbf{x}} \\ h_1 = \texttt{Dropout}_{0.5}(\texttt{Leaky}_{0.2}(\texttt{FC}_{N\to128}(h_0))) \\ h_2 = \texttt{Dropout}_{0.5}(\texttt{Leaky}_{0.2}(\texttt{FC}_{128\to128}(h_1))) \\ \mathcal{C}(\cdot) = FC_{128\to1}(h_2) \end{cases}$$

$\texttt{Dropout}_{0.5}$ is the dropout layer with probability 0.5. $\texttt{Leaky}_{0.2}$ is the LeakyReLU layer with 0.2 negative slope angle. `FC` is the fully connected linear layer with the corresponding input and output dimensions. $\texttt{gumbel}_{0.2}$ is the Gumbel-softmax activation function with scalar temperature 0.2. `tanh` represent `tanh` activation applied to the output of the linear layer.

3.5. Performance measure

The quality and interchangeability of synthetic data is assessed through different performance measures. The measures are statistical similarity, machine learning utility, and baseline privacy.

3.5.1. Statistical similarity

The similarity of the original and generated dataset is measured using pairwise correlation differences (PCD). PCD is the measure of difference of correlation matrices in terms of the Frobenius norm between real and synthetic data.

$$\text{PCD} = \|corr(\mathbf{X}_R) - corr(\mathbf{X}_S)\|_F \tag{23}$$

where $\mathbf{X}_R$ and $\mathbf{X}_S$ are the real and synthetic data matrices, *corr* is the correlation matrices of $\mathbf{X}_R$ and $\mathbf{X}_S$. PCD is computed using `dython`[1] package. The package provides a functionality that computes associations between numerical-numerical columns using Pearson's correlation. The Pearson correlation is given by

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{24}$$

For nominal-nominal or categorical-categorical associations, Crameŕ V is used to compute the associations. Crameŕ V is given by

$$\tilde{V} = \sqrt{\frac{\tilde{\varphi}^2}{\min(\tilde{k}-1, \tilde{r}-1)}}$$

where,

$$\tilde{\varphi}^2 = \max\left(0, \varphi^2 - \frac{(k-1)(r-1)}{n-1}\right) \tag{25}$$

and

$$\tilde{k} = k - \frac{(k-1)^2}{n-1}$$
$$\tilde{r} = r - \frac{(r-1)^2}{n-1}$$

$\varphi = \chi^2/n$ is the phi coefficient, χ is the Pearson's chi-squared test of the categorical variables from the contingency table, n is the total instances in the dataset, k is the number of columns, and r is the number of rows in the category. $\tilde{V}$ is the bias-corrected version of Crameŕ V.

For nominal-numerical columns, a correlation ratio is used. Correlation ratio (η) is defined as the curvilinear relationship between the statistical dispersion within the individual categories and the whole sample. In other words, it is the weighted variance of the mean of each category divided by the variance of the samples. The measure is the ratio of the standard deviations of the categories and the continuous sample. The correlation ratio intuitively answers the question: Given the continuous value, to which category does it belong?

$$\eta^2 = \frac{\sigma_{\bar{y}}^2}{\sigma_y^2}$$

where,

$$\begin{aligned} \sigma_{\bar{y}}^2 &= \frac{\sum_x n_x(\bar{y}_x - \bar{y})^2}{\sum_x n_x} \\ \sigma_y^2 &= \frac{\sum_{x,i}(y_{xi} - \bar{y})^2}{n} \\ \bar{y}_x &= \frac{\sum_i y_{xi}}{n_x} \\ \bar{y} &= \frac{\sum_x n_x \bar{y}_x}{\sum_x n_x} \end{aligned} \tag{26}$$

where y_{xi} is each observation with x that indicates the category the observation is in and i is the label of the particular observation. n_x is the number of observations in the category x. $\bar{y}_x$ is the mean of the observations in category x. $\bar{y}$ is the mean of all samples. PCD is measured at the dataset level. A smaller PCD value means better similarity between the generated data and the original dataset. When PCD is closer to 0, the artificial data is similar to the original dataset.

In addition, Kolmogorov–Smirnov test (KSTest) and Chi-Squared test (CSTest) are used to compare the distributions of the real and synthetic column. For continuous columns, KSTest is reported, which indicates the maximum distance between the expected and observed CDF. The empirical cumulative distribution function (CDF) of real and synthetic data distributions is compared using KSTest. For discrete columns, CSTest is reported, which computes the probability of real and synthetic data columns being sampled from the same distribution. Both these metrics are implemented using `Synthetic Data Vault (SDV)`[2] library.

3.5.2. Machine learning utility

The predictive ability between the real and synthetic data is assessed through machine learning utility. By using synthetic data for machine learning modeling, linear patterns and interactions similar to real data can be evaluated. This shows whether the synthetic data can be used as a proxy to replace the real data. Typically, the performance is tested by training the model on synthetic data and testing the model on real data as well as training the model on real data and testing the model on synthetic data.

Four different measures are considered to assess the machine learning utility between the real and synthetic data. The metrics are training using synthetic data and testing using real data (TSTR), training using real data and testing using synthetic data (TRTS), training using real data and testing using real data (TRTR), training using synthetic data and testing using synthetic data

[1] http://shakedzy.xyz/dython/

[2] https://sdv.dev/SDV/user_guides/evaluation/single_table_metrics.html

(TSTS). For classification datasets, F1-score is the main performance measure of choice to assess the machine learning utility. F1-score is the harmonic mean of precision and recall, which is given by

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{27}$$

where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{FP+FN}$. *TP* is the true positive, *FP* is the false positive, and *FN* is the false negative.

For regression datasets, root mean squared error (RMSE) error is the metric of choice. RMSE is the square root of the average squared difference between the observed value and the expected value. RMSE is given by

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{28}$$

where Y_i is the observed value, $\hat{Y}_i$ is the predicted value, and n is the number of data points.

3.5.3. Privacy

Privacy is quantified using two different distance-based measures. A distance-based measure is intuitive and easy to interpret the privacy of synthetic data. Distance to the closest record (DCR) and nearest neighbor distance ratio (NNDR) are the two distance-based privacy measures used in this research. DCR is the measure of Euclidean distance between any synthetic record and its nearest neighbor in the real dataset [4].

$$DCR(s_i, r_j) = \sum_{j=1}^{N} \|r_j - 1NN(s_i, r_{j\in n})\| \tag{29}$$

where, N is the size of the synthetic data, n is the size of the real data, s_i is the ith instance of the synthetic data, $r_{j\in n}$ is the real data, and $1NN(\cdot)$ is the first real data neighbor for the synthetic data s_i. The synthetic data is less private when the value of DCR is close to 0. In this case, the synthetic data contains an exact copy of the real data.

NNDR is the measure of ratio of the Euclidean distance between first and second nearest neighbors of synthetic data from the real data [24,32,33]. NNDR is defined by

$$NNDR(s_i, r_j) = \sum_{j=1}^{N} \frac{\|r_j - 1NN(s_i, r_{j\in n})\|}{\|r_j - 2NN(s_i, r_{j\in n})\|} \tag{30}$$

where, N is the size of the synthetic data, n is the size of the real data, s_i is the ith instance of the synthetic data, $r_{j\in n}$ is the real data, and $1NN(\cdot)$ and $2NN(\cdot)$ are the first and second neighbors of the synthetic data s_i from the real data. Typically, the value of NNDR is in the range [0, 1]. The privacy of the synthetic data is high when the value is closer to 1.

4. Experimental results

4.1. Datasets

The effectiveness of the proposed method is evaluated through an experimental setup. The datasets considered in this analysis are, in general, considered to be mainly for classification and regression tasks. The segregation of classification and regression tasks showcase the machine learning utility of the fake data. The datasets used in this research are taken from UCI machine learning repository [43] as well as from Faraway [44]. For classification tasks, *Mammographic mass* [43,45], *Caesarean* [43,46], *Cryotherapy* [43,47], *Immunotherapy* [43,47], and *Post-operative* [43] represent the datasets with low feature sizes. *Wisconsin Breast Cancer* [43,48–50], *Cleveland Heart Disease* [43], and *Cervical* [43,51] datasets are considered to have high feature sizes. To show the reliability of the proposed method, a high dimensional dataset *Urban Land Cover* [52,53] is considered for the analysis as well. All the above datasets have a binary target, except *Cleveland heart disease* data, which has a multi-class target type.

For regression tasks, *Thyroid* [43], *Liver* [43], and *Pima* [44] are considered to have low feature sizes. *Prostate* [44], *Fertility* [43,54], *Bioconcentration* [43,55,56], *Heart failure* [43,57], *Fat* [44], and *Parkinson's* [43,58] are considered to have high feature sizes. Similarly, a high dimensional dataset for regression, *Communities and Crime* [59] is considered for the analysis of regression tasks. The size of the datasets replicate small datasets in terms of number of rows. In addition to these benchmark datasets, a case-study data is considered one of the benchmark datasets [60]. The data represents a very small dataset size.

The proposed method handles mixed attribute types in the tabular data. To showcase this, the datasets for the experimental setup are chosen in a way that consist of discrete-only, continuous-only, and mixed attribute types. GenerativeMTD algorithmic steps can identify the discrete and continuous columns automatically. Appropriate losses are then computed for discrete-only and continuous-only columns based on the dataset distribution, respectively. In this case, only one of the terms in the reconstruction loss is computed in Eq. (21). For mixed attribute types, all three terms of Eq. (21) are computed.

Table 1 shows the summary of datasets. n represents the number of real data instances. p is the attribute size. $p_{discrete}$ and $p_{continuous}$ are the discrete and continuous column sizes in the real data, respectively. The procedure generates n_{obs} pseudo-real samples for each row and attribute sequentially. The total pseudo-real samples for each dataset are $n_{pseudo} = n_{obs} \times n \times 10$. Pseudo-real data is the input to generator. The best value of k is determined through sensitivity analysis by setting $k = [3, 10]$ similar to kNNMTD.

4.2. Experimental setup

The missing data rows are removed provided there are enough samples after removing the missing values. When necessary, the dataset is imputed for the missing values through Multivariate Imputation by Chained Equations (MICE) [61,62]. The categorical columns are one-hot encoded. The pseudo-real data is normalized based on their means and standard deviations. The procedure in Section 3.1 generates n_{obs} samples iteratively for each instance and attribute. The retained samples are concatenated column-wise. The algorithm generates n_{pseudo} number of pseudo-real data for training. Pseudo-real data is passed through the data loader to create mini-batch samples. The mini-batch sample size is set to real data's instance size n. This is because the real datasets' instance size is always as small as the mini-batch sample size for small datasets. A stochastic gradient descent (SGD) optimizer is used for both the networks with momentum ($\mu = 0.9$); weight decay for both the networks are set as $\lambda = 10^{-6}$. The learning rate used for both the networks is $\gamma = 2 \cdot 10^{-4}$. The two networks are trained simultaneously for 200 epochs. The gradient penalty is set as $\delta = 10$. Sinkhorn algorithm uses $\varepsilon = 0.01$ with $L = 100$ iterations. Number of critic iterations per generator training is set to $n_{critic} = 1$.

The final synthetic data sample size cannot be greater than n_{pseudo}. Because the final fake data is generated by sampling on the pseudo-real data for one epoch. The encoder and decoder are run for one epoch using the sampled pseudo-real data to generate the final synthetic data. Primary performance measure of choice to select the best model during model training is PCD. The best network models are saved based on the best PCD value encountered while training. The final fake data is then inverse-transformed

Table 1
Summary of the benchmark datasets.

Task	Data	n	p	$p_{discrete}$	$p_{continuous}$	n_{obs}	n_{pseudo}[a]
Classification	*Mammographic mass*	830	6	5	1	100	830 000
	Caesarean	80	6	1	5	100	80 000
	Cryotherapy	90	7	3	4	100	90 000
	Immunotherapy	90	8	3	5	100	90 000
	Post-operative	86	9	9	–	100	86 000
	Wisconsin breast cancer	683	10	9	1	100	683 000
	Cleveland heart disease	297	14	9	5	10	29 700
	Cervical	72	20	20	–	100	72 000
	Sweat Study	48	19	5	14	100	48 000
	Urban land cover	675	148	–	148	10	67 500
Regression	*Thyroid*	215	6	2	4	100	215 000
	Liver	345	6	5	1	100	345 000
	Pima	768	9	7	2	100	768 000
	Prostate	97	9	4	5	100	97 000
	Fertility	100	10	6	4	100	100 000
	Bioconcentration	779	11	5	6	100	779 000
	Heart failure	229	13	10	3	100	229 000
	Fat	252	17	1	16	100	252 000
	Parkinson's	195	23	1	22	100	195 000
	Sweat Study (Regression)	48	19	5	14	100	48 000
	Communities and Crime	319	123	2	121	10	31 900

[a] $n_{pseudo} = n \times n_{obs} \times 10$.

into the original tabular form using the real and pseudo-real dataset's combined means and standard deviations. The generated final fake data is then used to compute the statistical measures, utility metrics, and privacy metrics.

VEEGAN [23], TableGAN [4], CTGAN [1], and TVAE [1] are the benchmark models considered to compare our proposed method. The code for the benchmark models is publicly available in SDGym.[3] The benchmark datasets are fit using these methods. Each of the benchmark models is trained for 200 epochs similar to GenerativeMTD. The final fake data from these models are then sampled based on the fitted model. The final samples are then used to compute the performance metrics similar to GenerativeMTD.

4.3. Results

A sensitivity analysis is conducted to study the influence of k (in Section 3.1) on PCD for GenerativeMTD. Figs. 2 and 3 show the effect of k on PCD for all the benchmark datasets. To identify the best synthetic data for a specific dataset, performing sensitivity analysis will help decide the best value of k. For some k values, GenerativeMTD achieves the lowest PCD value for the benchmark datasets. The best k value for a dataset depends on the synthetic data goal. If the goal is to achieve low PCD, choosing the appropriate k that has the lowest PCD value through sensitivity analysis is feasible.

When the PCD is high for a dataset, the synthetic data generated is farther from the real data. In such cases, DCR and NNDR are expected to be high. When PCD is high, the privacy of the synthetic data is expected to be high, which implies noisy or irrelevant data. Additionally, the machine learning utility will be low in such a scenario. This is the reason why synthetic datasets with lowest PCD are selected first and then the best NNDR value to achieve better privacy. The synthetic data with the lowest PCD is chosen first. Then the synthetic data with best NNDR is chosen second. GenerativeMTD consistently performs better in terms of PCD and NNDR compared to other benchmark models in Tables 2 and 3. The arrows indicate whether a higher value or a lower value is ideal for the corresponding metric. The NNDR values of GenerativeMTD, although not the best, are still closer to the best competing benchmark model for all the datasets. Experimentally, GenerativeMTD is able to achieve better statistical similarity and achieve privacy. Because the privacy metrics are higher, the machine learning utility is not effective. GenerativeMTD's machine learning utility will be improved greatly after incorporating the final step of kNNMTD from our previous work. The final step of kNNMTD adopted from Sivakumar et al. [7] under GenerativeMTD's process is currently ignored in this study. Adding this final step will hugely affect the performance of synthetic data. Alternatively, ignoring the final step does affect the privacy metrics as well. To avoid creating copies of real data, this step is ignored in GenerativeMTD. GenerativeMTD's main objective is quality and privacy-preserving synthetic data.

CTGAN consistently gave better CSTest values for all the datasets. This is expected because the architecture takes conditional categorical vectors as an input. GenerativeMTD sometimes comes close to CTGAN's value (e.g., *Cryotherapy, sweat study*). VEEGAN shows better privacy, but that is also not trustworthy due to high PCD value (e.g., *Parkinsons*). TableGAN shows better privacy, but GenerativeMTD comes close as well (e.g., *Cryotherapy*). For regression datasets, only KSTest values seem to be better for TableGAN. TVAE show better KSTest, TSTR, and TRTS for both classification and regression datasets. The benchmark model performance is highly data specific. Each of these methods is designed to handle tabular datasets but without imposing any constraints on statistical similarity, machine learning utility, and privacy. GenerativeMTD is designed to optimize on statistical similarity and privacy.

kNNMTD and GenerativeMTD are synthetic data generation algorithms. Random Forest classifier and regressor are used just to showcase the machine learning utility of the synthetic data generated by benchmark methods and GenerativeMTD. In the absence of pseudo-real data, training the deep learning model that uses small real datasets is not feasible. After applying GenerativeMTD, the resulting synthetic data can be further used for any data-driven modeling. The experimental setup is designed to illustrate synthetic data generation as a preprocessing step in a traditional data modeling pipeline for small datasets. Additionally, a post-processing step to utilize the generated fake data for classification/regression model training is also illustrated. This similar setup is only limited to tabular data but any other input data types as well.

[3] https://github.com/sdv-dev/SDV

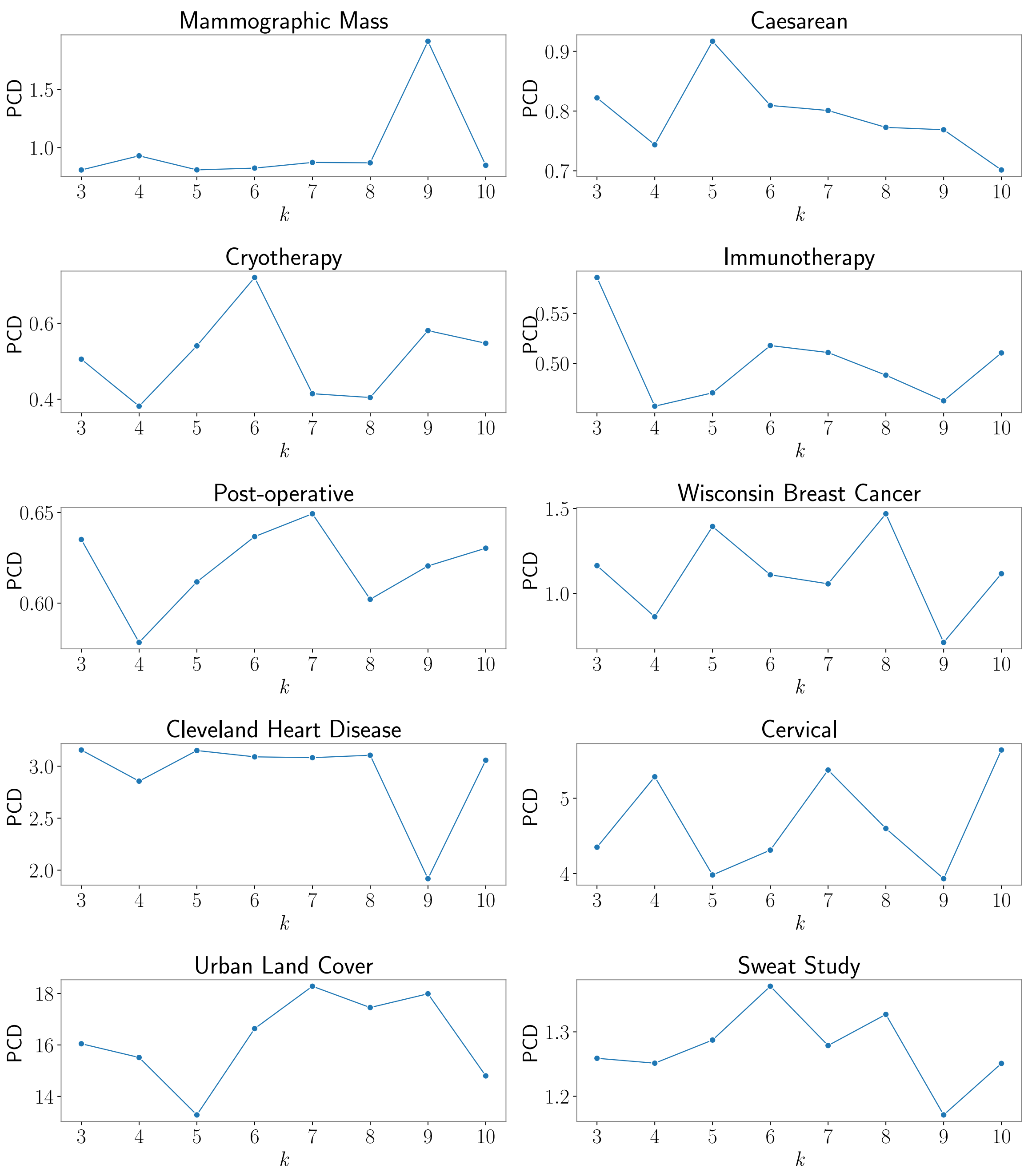


Fig. 2. Effect of k on PCD of GenerativeMTD for the classification benchmark datasets.

Machine learning utility is data specific. The experimental setup illustrates only the usage of this utility metric. The best machine learning model that shows high utility metric is currently out of the scope. In the future, the machine learning model that improves the TSTR & TRTS metric using the currently generated synthetic datasets will be explored. Hyperparameter tuning can be utilized in the future to improve the learning accuracies. But to understand if the synthetic data generated is a better proxy, the machine learning utility metrics are reported. When TSTR and TRTS values are better than TRTR, we can infer that the synthetic datasets are a better proxy for real datasets.

When there is a trade-off between PCD and NNDR, the machine learning utility is affected. Similar to machine learning utility, this pattern is observed for KSTest as well as CSTest. A common ground among these metrics will provide a quality and privacy-preserving synthetic data with high utility for any real data. Tables 4 and 5 compare the parent sampling algorithm *k*NNMTD and GenerativeMTD. In both these tables, PCD is better for *k*NNMTD for all datasets. The machine learning utility is also better for *k*NNMTD. This comparison proves the claim that the better PCD value increases the machine learning utility but reduces data privacy. But GenerativeMTD's PCD for all datasets is around the range of *k*NNMTD's PCD values and the proposed method has better NNDR values, which illustrates better synthetic data quality without giving up privacy. The proposed method can generate quality synthetic data without incorporating the final

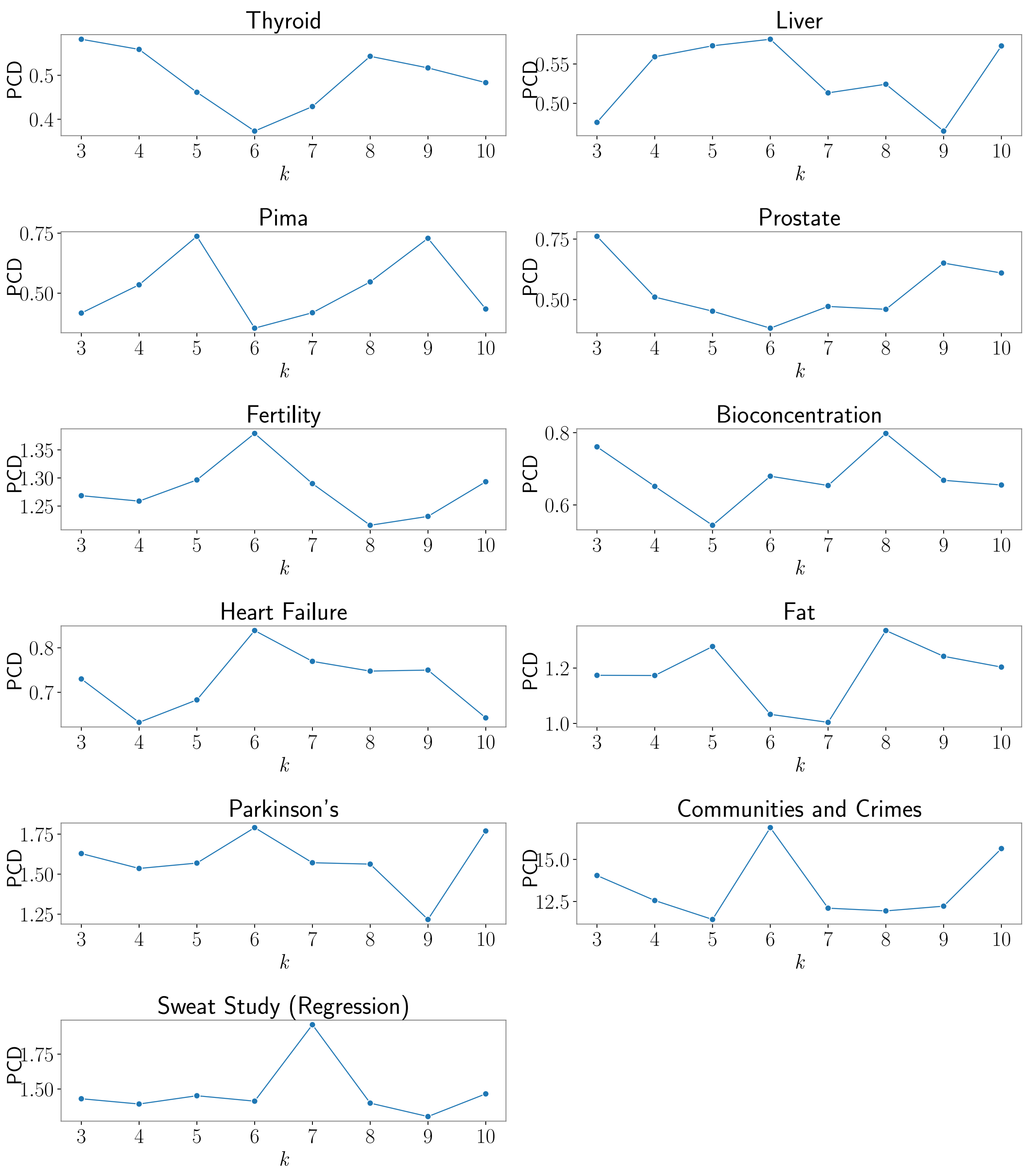


Fig. 3. Effect of *k* on PCD of GenerativeMTD for the regression benchmark datasets.

step from *k*NNMTD. The machine learning utility is believed to be better after hyperparameter tuning. The PCD curve for each epoch is shown in the Supplementary Material attached to this study. GenerativeMTD is capable of optimizing on PCD as well as the distance-based privacy metric together.

GenerativeMTD tackles mixed attribute types through its parent algorithm *k*NNMTD. *k*NNMTD synthesizes data based on its real dataset type. Therefore, pseudo-real data retains the real dataset type under GenerativeMTD. Data fidelity needs more investigation. The sequential nature of *k*NNMTD for pseudo-real data is generated around the vicinity of the real data. *k*NNMTD's unmodified version generate balanced pseudo-real data. But in this setup, the algorithm chooses the class distribution randomly. GenerativeMTD can be tweeked to generate balanced pseudo-real data. GenerativeMTD is not domain specific as illustrated in the experimental setup. It requires no user intervention and provides quality synthetic data in terms of statistical similarity and privacy. GenerativeMTD handles only tabular (structured) data. GenerativeMTD's ability to handle text, audio, and image inputs are out of the scope of this research. In the future, time-series attributes under structured data will be explored. A thorough analysis on data fidelity and membership attacks for the synthetic data generated through *k*NNMTD and GenerativeMTD will be explored.

One of the limitations of GenerativeMTD is the running time. *k*NNMTD and GenerativeMTD (pseudo-real data generation step)

Table 2
Performance metrics of synthetic data from the proposed and benchmark methods for classification datasets.

Data	Method	*k*	PCD	KSTest	CSTest	TSTR	TRTS	TRTR	TSTS	DCR	NNDR
			(↓)	(↑)	(↑)	F1-Score (↑)				(↑)	(↑)
Mammographic mass	VEEGAN	–	1.2744	0.7280	0.7920	0.7753	0.7556	0.7948	0.8175	1.8754 ± 0.9402	0.8363 ± 0.2192
	TableGAN	–	1.4243	0.9520	0.3274	0.7610	0.5230	0.7901	0.6503	1.5002 ± 0.5151	**0.8707 ± 0.2091**
	TVAE	–	1.2652	**0.9625**	**0.9857**	0.8265	**0.8376**	0.7926	0.8616	0.3096 ± 0.5711	0.2598 ± 0.4266
	CTGAN	–	1.9940	0.6124	0.9771	0.5894	0.5707	**0.7974**	0.5731	2.5868 ± 3.2718	0.8032 ± 0.2830
	GenerativeMTD	3	**0.8079**	0.3615	0.7871	**0.8329**	0.8257	0.7942	**0.9517**	**18.5516 ± 18.8904**	0.8061 ± 0.2577
Caesarean	VEEGAN	–	0.8917	0.7935	0.8506	0.5460	0.4901	0.4842	0.4700	1.4565 ± 0.5707	0.7748 ± 0.2443
	TableGAN	–	0.9354	0.8475	0.6459	0.4359	**0.5576**	**0.4854**	0.5000	1.0568 ± 0.4879	0.7640 ± 0.3337
	TVAE	–	1.8024	**0.8855**	0.6881	0.0547	0.2373	0.4343	0.1445	1.0084 ± 0.4161	0.7834 ± 0.3154
	CTGAN	–	0.9120	0.8010	**0.9915**	0.4137	0.3641	0.4372	0.4479	1.6177 ± 0.8468	0.7937 ± 0.2302
	GenerativeMTD	3	**0.8221**	0.4115	0.9467	**0.6580**	0.5138	0.4199	**0.7079**	**8.8129 ± 8.4501**	**0.8009 ± 0.2215**
Cryotherapy	VEEGAN	–	0.9293	0.6647	0.9202	0.6000	0.7227	0.9167	0.8033	7.7575 ± 4.0980	0.8782 ± 0.131
	TableGAN	–	1.1935	0.6699	0.3268	0.7064	0.5414	**0.9449**	0.6474	**85.4450 ± 85.7248**	**0.9747 ± 0.0606**
	TVAE	–	0.9819	0.7975	0.9025	**0.8538**	0.7535	0.9449	0.793	9.1656 ± 5.4979	0.911 ± 0.1177
	CTGAN	–	1.7566	0.7947	**0.9822**	0.5484	0.4162	0.9316	0.4862	49.9916 ± 92.0925	0.9575 ± 0.0731
	GenerativeMTD	10	**0.5473**	**0.8071**	0.9638	0.8513	**0.9267**	0.9317	**0.9848**	13.3684 ± 8.1907	0.9420 ± 0.0860
Immunotherapy	VEEGAN	–	1.0809	0.6622	0.8086	0.3733	0.2135	0.3633	0.5003	15.4114 ± 7.8537	0.8289 ± 0.1465
	TableGAN	–	0.9011	0.6488	0.2656	0.0190	0.1790	0.3633	0.0508	**25.8453 ± 19.0666**	0.7416 ± 0.1947
	TVAE	–	1.7964	**0.7631**	0.6521	0.0000	**0.9696**	**0.4633**	**1.0000**	7.7946 ± 3.3073	0.7698 ± 0.1708
	CTGAN	–	0.9367	0.6066	**0.9331**	0.1303	0.0863	0.3633	0.1393	23.7620 ± 36.2700	**0.8716 ± 0.1384**
	GenerativeMTD	6	**0.5178**	0.7037	0.9003	**0.6488**	0.0300	0.4433	0.6522	20.4425 ± 9.0781	0.8190 ± 0.1805
Post-operative	VEEGAN	–	1.7321	–	0.8363	0.8051	0.8034	0.7761	0.934	0.6132 ± 0.5540	0.5491 ± 0.4827
	TableGAN	–	0.7359	–	0.6371	0.6807	0.6912	0.7742	0.7821	**3.0802 ± 0.3680**	**0.8841 ± 0.0841**
	TVAE	–	2.0607	–	0.8629	**0.8358**	**0.8546**	0.7696	**0.9863**	0.3779 ± 0.4881	0.3694 ± 0.4777
	CTGAN	–	0.6078	–	**0.9677**	0.7526	0.7371	**0.7897**	0.7142	1.0946 ± 0.6260	0.7263 ± 0.3664
	GenerativeMTD	8	**0.6021**	–	0.6614	0.6488	0.6101	0.7833	0.4805	1.8244 ± 0.5809	0.8230 ± 0.1784
Wisconsin breast cancer	VEEGAN	–	2.8994	0.7043	0.6648	0.9254	0.7231	**0.9763**	0.839	4.3937 ± 1.7090	0.8663 ± 0.1876
	TableGAN	–	3.0631	**0.8039**	0.4800	**0.9652**	0.9129	0.9752	0.9279	3.0227 ± 1.4009	0.8872 ± 0.1619
	TVAE	–	2.5252	0.6408	0.8999	0.9139	**0.9631**	0.9763	0.9760	2.1018 ± 0.8505	**0.9364 ± 0.1182**
	CTGAN	–	5.9651	0.7034	0.776	0.1301	0.3913	0.9763	0.2426	**6.6389 ± 1.7588**	0.9122 ± 0.0859
	GenerativeMTD	8	**1.4689**	0.7034	**0.9740**	0.9158	0.948	0.9752	**0.9868**	4.6029 ± 1.8657	0.9141 ± 0.0859
Cleveland heart disease	VEEGAN	–	2.9316	0.6776	0.9093	**0.5980**	0.6656	0.5656	0.8640	36.4280 ± 14.7451	0.8462 ± 0.1396
	TableGAN	–	4.1148	**0.8998**	0.4744	0.4027	0.3242	0.5658	0.4440	21.2139 ± 6.0467	0.8904 ± 0.0991
	TVAE	–	2.8585	0.7782	0.9016	0.5717	**0.9208**	0.5690	**0.9500**	14.5800 ± 4.0868	0.8482 ± 0.1242
	CTGAN	–	4.7294	0.6755	**0.9736**	0.5158	0.3638	**0.5723**	0.3980	35.7454 ± 15.0510	0.8932 ± 0.0970
	GenerativeMTD	9	**1.9183**	0.5984	0.7410	0.5556	0.4428	0.5555	0.4840	**76.0260 ± 44.3935**	**0.9029 ± 0.1028**
Cervical	VEEGAN	–	6.4445	0.7299	0.8910	0.8386	0.9228	0.9240	**0.9814**	8.6029 ± 2.1812	0.9326 ± 0.0598
	TableGAN	–	4.9770	0.6829	0.1222	0.8884	0.7560	0.9023	0.8865	11.2232 ± 1.7347	0.9159 ± 0.0641
	TVAE	–	**3.6385**	**0.8240**	0.9725	**0.9137**	**0.9262**	0.9023	0.9619	8.5232 ± 2.1838	0.8926 ± 0.0905
	CTGAN	–	6.7498	0.7174	**0.9998**	0.8265	0.7153	0.9153	0.8245	15.7518 ± 2.8540	0.9426 ± 0.0537
	GenerativeMTD	8	4.5987	0.5717	0.9246	0.7913	0.8902	**0.9240**	0.9671	**17.8860 ± 5.6221**	**0.9551 ± 0.0530**
Urban land cover	VEEGAN	–	58.8566	0.5658	0.9563	0.2290	0.2816	0.8652	0.7850	6617.1869 ± 1862.6574	**0.9834 ± 0.0159**
	TableGAN	–	55.2505	**0.90000**	0.9882	0.1816	0.1518	0.8696	0.6530	4520.266 ± 2549.2089	0.9429 ± 0.0581
	TVAE	–	15.5274	0.8723	1.0000	**0.7093**	**0.7436**	0.8696	**0.8310**	2551.3755 ± 927.9601	0.9330 ± 0.0642
	CTGAN	–	52.4371	0.8448	1.0000	0.1393	0.1240	**0.8726**	0.156	**7340.3822 ± 4225.2489**	0.9605 ± 0.0600
	GenerativeMTD	10	**14.7979**	0.6889	**1.0000**	0.6836	0.5838	0.8652	0.814	3688.9947 ± 1225.1702	0.9265 ± 0.0822
Sweat study (case study)	VEEGAN	–	3.7839	0.7003	0.7627	0.7848	0.4662	**0.7429**	**1.0000**	826.3304 ± 619.4761	0.8059 ± 0.1757
	TableGAN	–	6.4385	**0.8083**	0.5592	0.7240	0.7744	0.7333	0.8852	718.4314 ± 426.0045	0.8761 ± 0.1108
	TVAE	–	4.2106	0.8007	0.8138	**0.7855**	**0.8897**	0.7048	0.9712	445.4728 ± 182.1186	0.8595 ± 0.1227
	CTGAN	–	6.3741	0.6482	**0.9599**	0.7786	0.6218	0.7538	0.7994	**1450.6842 ± 977.0765**	**0.8986 ± 0.1105**
	GenerativeMTD	4	**1.2514**	0.7370	0.9423	0.7306	0.7129	0.7341	0.8532	341.2201 ± 96.6132	0.8639 ± 0.1177

Best synthetic data for comparison is chosen based on PCD and NNDR.

are both exclusively designed to tackle small datasets. The sequential procedure for synthesis is not optimized for big data. Irrespective of the time complexity, the algorithm works for large datasets. But the running time will be improved in the future to incorporate faster training. Table 6 shows the total running time (in minutes) for the experimental setup with 200 epochs. The loss curves and the distributions for each of these datasets are provided in the supplementary material.

5. Conclusion

GenerativeMTD outperforms all the existing benchmark models for tabular data generation in terms of quality and privacy. The synthetic data from the proposed method is statistically similar to real data with best PCD values. It represents a good proxy of real data in terms of machine learning utilities and achieves baseline privacy in terms of distance metrics. GenerativeMTD is the only existing synthetic data generation framework that handles small datasets as well as generates realistic data for any dataset size. The synthetic data from GenerativeMTD is a promising surrogate that can be used efficiently for research and analysis purposes.

GenerativeMTD handles only numerical data. The non-numeric categorical levels need to be one-hot encoded before generating synthetic data. GenerativeMTD inverse-transforms the one-hot encoded columns to their original form in the output synthetic data. The pseudo-real samples generation step is specifically designed to handle the small datasets. Therefore, handling very large datasets using GenerativeMTD is a time-consuming process. In the future, any type of tabular data, such as time-series or text data will be handled by GenerativeMTD. A very efficient and faster computational methodology will be implemented for pseudo-real data generation for large datasets. In addition, an attempt to use images as an input will be explored in the future. The membership and attribute attacks will be tested on the synthetic data generated from GenerativeMTD.

Table 3
Performance metrics of synthetic data from the proposed and benchmark methods for regression datasets.

Data	Method	*k*	PCD	KSTest	CSTest	TSTR	TRTS	TRTR	TSTS	DCR	NNDR
			(↓)	(↑)	(↑)	RMSE Score (↓)				(↑)	(↑)
Thyroid	VEEGAN	–	2.1822	0.3118	0.1384	7.4599	20.1555	5.1144	13.5852	9.8947 ± 6.8087	0.8683 ± 0.1210
	TableGAN	–	2.4426	0.5772	0.8158	7.0776	19.1836	4.8796	5.9531	1.6900 ± 5.1098	0.7988 ± 0.0989
	TVAE	–	1.4660	**0.8330**	0.8863	**6.5390**	**2.6133**	4.9734	**1.9974**	10.8945 ± 1.2361	0.8545 ± 0.1608
	CTGAN	–	2.7928	0.6176	0.9369	9.5007	10.2699	**4.8521**	3.8135	8.3650 ± 7.9504	0.8338 ± 0.1158
	GenerativeMTD	6	**0.3731**	0.7329	**0.9848**	6.7934	6.6397	4.9925	3.5892	$\mathbf{33.4984 \pm 31.9310}$	$\mathbf{0.9002 \pm 0.1211}$
Liver	VEEGAN	–	2.7530	0.4877	–	21.6854	32.3905	**19.9547**	25.3194	17.9224 ± 23.0453	0.8662 ± 0.0947
	TableGAN	–	1.5972	**0.8237**	–	**19.5485**	19.4317	20.2226	17.4408	6.8363 ± 5.5356	0.8405 ± 0.1215
	TVAE	–	1.2496	0.7464	–	23.9204	**16.2572**	20.0436	**14.8585**	33.6845 ± 2.3103	0.8872 ± 0.1339
	CTGAN	–	1.8644	0.7161	–	22.1556	27.3906	20.0610	24.6468	40.6708 ± 9.3709	0.8095 ± 0.1196
	GenerativeMTD	8	**0.5243**	0.6617	–	21.3937	34.6077	20.1129	34.1588	$\mathbf{66.4342 \pm 50.6624}$	$\mathbf{0.9236 \pm 0.1080}$
Pima	VEEGAN	–	1.8013	0.6435	0.8323	0.3947	0.5904	**0.3351**	0.4765	29.5440 ± 15.1700	0.8926 ± 0.1170
	TableGAN	–	2.3494	**0.8156**	**0.9549**	0.3648	0.3196	0.3382	0.2364	11.7238 ± 9.0064	0.8555 ± 0.1144
	TVAE	–	1.1806	0.7955	0.8192	0.3559	**0.2243**	0.338	**0.1177**	32.5860 ± 4.4061	0.8845 ± 0.1245
	CTGAN	–	1.8869	0.7124	0.8272	0.3879	0.4851	0.3368	0.2589	27.4080 ± 19.8523	0.8711 ± 0.1018
	GenerativeMTD	9	**0.7286**	0.6310	0.9397	**0.3312**	0.5121	0.3375	0.2515	$\mathbf{57.8178 \pm 27.8601}$	$\mathbf{0.9285 \pm 0.0749}$
Prostate	VEEGAN	–	1.5526	0.6534	0.4696	1.2041	1.9711	0.7713	1.5527	10.7637 ± 2.2310	$\mathbf{0.9162 \pm 0.1478}$
	TableGAN	–	3.3172	0.6981	0.4676	1.2104	1.3047	**0.7651**	1.0770	3.7616 ± 2.1752	0.8072 ± 0.0854
	TVAE	–	1.6058	**0.8341**	0.8230	0.8698	0.8985	0.7737	0.8229	5.2168 ± 1.8031	0.8290 ± 0.1701
	CTGAN	–	3.2977	0.5577	**0.8928**	1.8755	2.1127	0.7671	1.5531	7.0235 ± 6.4752	0.8249 ± 0.1104
	GenerativeMTD	10	**0.6100**	0.7262	0.8073	**0.8189**	**0.4771**	0.7700	**0.2928**	$\mathbf{11.2659 \pm 8.0597}$	0.7964 ± 0.1739
Fertility	VEEGAN	–	**1.2092**	0.7600	0.8875	0.2504	0.2860	0.1783	0.2722	0.7522 ± 0.3276	0.7726 ± 0.2404
	TableGAN	–	1.4695	**0.8665**	0.4971	0.2282	0.2163	**0.1791**	0.1918	0.2003 ± 0.3791	0.6092 ± 0.2075
	TVAE	–	2.2814	0.8315	0.7797	**0.1752**	**0.1342**	0.1802	**0.1197**	0.4507 ± 0.1096	0.6772 ± 0.2318
	CTGAN	–	1.4309	0.4965	**0.9792**	0.2712	0.3841	0.1815	0.2125	0.6434 ± 0.3746	0.7241 ± 0.2235
	GenerativeMTD	3	1.2685	0.4350	0.7216	0.4431	0.3890	0.1796	0.2044	$\mathbf{1.1027 \pm 0.4084}$	$\mathbf{0.8380 \pm 0.1756}$
Bioconcentration	VEEGAN	–	2.3079	0.5880	**0.9354**	1.4805	2.3176	0.632	1.8352	$\mathbf{3.4541 \pm 1.0682}$	$\mathbf{0.9300 \pm 0.0819}$
	TableGAN	–	3.0572	0.7485	0.6439	1.1921	1.2431	0.6334	0.9816	1.3659 ± 0.6965	0.8728 ± 0.0678
	TVAE	–	1.3076	**0.7930**	0.9266	0.8962	1.1404	0.6339	0.9720	3.2889 ± 0.6652	0.9225 ± 0.1323
	CTGAN	–	3.0098	0.6999	0.9214	1.9917	2.7536	0.6330	1.6814	3.3532 ± 1.2173	0.9221 ± 0.0736
	GenerativeMTD	10	**0.6551**	0.7722	0.6272	**0.8420**	**0.5262**	**0.6305**	**0.3686**	2.3936 ± 0.8188	0.9037 ± 0.0968
Heart failure	VEEGAN	–	1.5657	0.6088	0.9100	6008.6561	5376.1362	1006.1709	478.6972	2836.5078 ± 12831.1335	0.6209 ± 0.2665
	TableGAN	–	2.2652	0.7350	0.8912	1847.1691	1814.5654	1019.8672	758.7700	699.4358 ± 1530.0717	0.6607 ± 0.2439
	TVAE	–	1.6547	**0.7555**	0.5618	1055.6600	**838.4858**	**1004.9518**	**118.7596**	$\mathbf{12765.6953} \pm 660.0031$	0.6996 ± 0.2977
	CTGAN	–	1.4480	0.6742	0.9391	**985.7872**	1184.8776	1011.1480	999.5736	2596.7926 ± 6310.2768	0.5880 ± 0.2858
	GenerativeMTD	5	**0.6829**	0.5639	**0.9619**	994.8289	1342.6452	1012.9494	586.8402	1262.8066 ± 972.5670	$\mathbf{0.7118 \pm 0.2584}$
Fat	VEEGAN	–	4.7598	0.5587	–	5.0649	11.7026	1.3619	9.1002	33.6176 ± 20.8100	0.9492 ± 0.1099
	TableGAN	–	7.3044	**0.9006**	–	7.4077	7.7588	1.3389	**2.2887**	16.1783 ± 5.9254	0.9243 ± 0.0750
	TVAE	–	5.5590	0.8213	–	4.0071	6.9497	1.3284	5.7031	43.8734 ± 5.4989	0.9253 ± 0.0707
	CTGAN	–	9.9813	0.7068	–	7.6658	14.5288	**1.3064**	9.0286	36.5204 ± 11.2576	0.9531 ± 0.0478
	GenerativeMTD	6	**1.0328**	0.7090	–	**3.4595**	**3.2099**	1.3504	2.7956	$\mathbf{47.3405 \pm 21.619}$	$\mathbf{0.9608 \pm 0.0420}$
Parkinson's	VEEGAN	–	5.5842	0.5346	0.7543	0.0824	0.1294	0.0219	0.0874	$\mathbf{73.2176 \pm 33.7600}$	$\mathbf{0.8898 \pm 0.1312}$
	TableGAN	–	5.8799	**0.8473**	**0.9429**	**0.0265**	0.0319	0.0226	0.0293	22.4596 ± 27.7607	0.8567 ± 0.1529
	TVAE	–	8.2752	0.7830	0.6142	0.0719	0.0679	**0.0220**	0.0506	56.8349 ± 17.3232	0.9060 ± 0.1503
	CTGAN	–	12.7516	0.6859	0.4022	0.1313	0.1576	0.0226	0.0941	38.5789 ± 64.3387	0.8748 ± 0.1372
	GenerativeMTD	9	**1.2171**	0.8329	0.8221	0.0426	**0.0219**	0.0224	**0.0169**	26.5523 ± 14.9309	0.8204 ± 0.1703
Communities and Crimes	VEEGAN	–	40.1628	0.4753	0.8212	0.2045	**0.1775**	0.1648	**0.0771**	2.6537 ± 0.1658	0.9845 ± 0.0105
	TableGAN	–	47.4388	**0.8542**	0.0757	0.2376	0.2308	**0.1642**	0.1347	1.4600 ± 0.2998	0.9572 ± 0.0267
	TVAE	–	19.9041	0.8024	0.9259	**0.1840**	0.1954	0.1646	0.1584	2.5172 ± 0.2306	$\mathbf{0.9862 \pm 0.0386}$
	CTGAN	–	37.6413	0.8071	**0.9453**	0.3181	0.3234	0.1649	0.2897	2.8139 ± 0.2566	0.9837 ± 0.0141
	GenerativeMTD	6	**16.8932**	0.5846	0.3449	0.1971	0.2927	0.1654	0.1675	$\mathbf{3.0106 \pm 0.2590}$	0.9724 ± 0.0217
Sweat study (case study)	VEEGAN	–	5.2890	0.6142	0.7626	1.0583	0.8125	0.8360	**0.3143**	$\mathbf{1196.8658 \pm 635.9900}$	$\mathbf{0.9104 \pm 0.1357}$
	TableGAN	–	7.3352	**0.8067**	0.5574	0.9112	**0.7393**	**0.8265**	0.4623	457.5061 ± 377.0329	0.8762 ± 0.1167
	TVAE	–	4.0139	0.8008	0.8352	0.9611	0.8081	0.8318	0.3551	809.9112 ± 191.5558	0.8688 ± 0.1171
	CTGAN	–	6.3776	0.6761	**0.9751**	0.8231	0.8320	0.8344	0.7898	1100.6563 ± 676.2290	0.9071 ± 0.0829
	GenerativeMTD	5	**1.4521**	0.7414	0.9564	**0.6936**	0.7847	0.8296	0.6012	330.8251 ± 92.3862	0.8528 ± 0.1316

Best synthetic data for comparison is chosen based on PCD and NNDR.

CRediT authorship contribution statement

Jayanth Sivakumar: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Karthik Ramamurthy:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Conceptualization. **Menaka Radhakrishnan:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Conceptualization. **Daehan Won:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Funding acquisition, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code is available on https://github.com/jsivaku1/GenerativeMTD.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2023.110956.

Table 4
Synthetic data performance of GenerativeMTD and *k*NNMTD for classification datasets.

Data	Method	*k*	PCD	KSTest	CSTest	TSTR	TRTS	TRTR	TSTS	DCR	NNDR
			(↓)	(↑)	(↑)	F1-Score (↑)				(↑)	(↑)
Mammographic mass	*k*NNMTD	10	**0.3662**	**0.9440**	**0.9826**	**0.9687**	**0.9474**	0.7561	0.9473	0.0020 ± 0.0140	0.0008 ± 0.0057
	GenerativeMTD	3	0.8079	0.3615	0.7871	0.8329	0.8257	**0.7942**	**0.9517**	$\mathbf{18.5516 \pm 18.8904}$	$\mathbf{0.8061 \pm 0.2577}$
Caesarean	*k*NNMTD	8	**0.4624**	**0.9640**	**0.9845**	**0.9277**	**0.8958**	**0.5164**	**0.8854**	0.0000 ± 0.0000	0.0000 ± 0.0000
	GenerativeMTD	3	0.8221	0.4115	0.9467	0.6580	0.5138	0.4199	0.7079	$\mathbf{8.8129 \pm 8.4501}$	$\mathbf{0.8009 \pm 0.2215}$
Cryotherapy	*k*NNMTD	8	**0.3008**	**0.9470**	**0.9654**	**0.9892**	**0.9728**	0.8370	0.9682	0.0033 ± 0.0075	0.0168 ± 0.0788
	GenerativeMTD	10	0.5473	0.8071	0.9638	0.8513	0.9267	**0.9317**	**0.9848**	$\mathbf{13.3684 \pm 8.1907}$	$\mathbf{0.9420 \pm 0.0860}$
Immunotherapy	*k*NNMTD	6	**0.2819**	**0.9662**	**0.9917**	**0.9948**	**0.9244**	**0.5782**	**0.9873**	0.0031 ± 0.0044	0.0002 ± 0.0004
	GenerativeMTD	6	0.5178	0.7037	0.9003	0.6488	0.0300	0.4433	0.6522	$\mathbf{20.4425 \pm 9.0781}$	$\mathbf{0.8190 \pm 0.1805}$
Post-operative	*k*NNMTD	10	0.7907	–	**0.9888**	**0.8989**	**0.8318**	0.4142	**0.8235**	0.0000 ± 0.0000	0.0000 ± 0.0000
	GenerativeMTD	8	**0.6021**	–	0.6614	0.6488	0.6101	**0.7833**	0.4805	$\mathbf{1.8244 \pm 0.5809}$	$\mathbf{0.8230 \pm 0.1784}$
Wisconsin breast cancer	*k*NNMTD	7	**0.7379**	**0.9426**	**0.9877**	**0.9970**	**0.9952**	0.9710	**0.9979**	0.0000 ± 0.0000	0.0000 ± 0.0000
	GenerativeMTD	8	1.4689	0.7034	0.9740	0.9158	0.9480	**0.9752**	0.9868	$\mathbf{4.6029 \pm 1.8657}$	$\mathbf{0.9141 \pm 0.0859}$
Cleveland heart disease	*k*NNMTD	3	6.1577	**0.9350**	0.1667	**0.8369**	**0.8478**	0.2323	**0.7160**	0.2068 ± 0.3203	0.0109 ± 0.0199
	GenerativeMTD	9	**1.9183**	0.5984	**0.7410**	0.5556	0.4428	**0.5555**	0.4840	$\mathbf{76.0260 \pm 44.3935}$	$\mathbf{0.9029 \pm 0.1028}$
Cervical	*k*NNMTD	6	**2.0701**	**0.9714**	**1.0000**	**0.9971**	**0.9685**	0.8111	**0.9845**	0.0000 ± 0.0000	0.0000 ± 0.0000
	GenerativeMTD	8	4.5987	0.5717	0.9246	0.7913	0.8902	**0.9240**	0.9671	$\mathbf{17.8860 \pm 5.6221}$	$\mathbf{0.9551 \pm 0.0530}$
Urban land cover	*k*NNMTD	6	**11.6063**	**0.9280**	**1.0000**	**0.8608**	**0.9700**	0.8488	**0.8300**	2.1621 ± 3.1217	0.0010 ± 0.0017
	GenerativeMTD	10	14.7979	0.6889	**1.0000**	0.6836	0.5838	**0.8652**	0.8140	$\mathbf{3688.9947 \pm 1225.1702}$	$\mathbf{0.9265 \pm 0.0822}$
Sweat study (case study)	*k*NNMTD	7	**0.9485**	**0.9290**	0.9367	**0.9693**	**0.9083**	0.6607	**0.9351**	1.2010 ± 2.5877	0.0030 ± 0.0033
	GenerativeMTD	4	1.2514	0.7370	**0.9423**	0.7306	0.7129	**0.7341**	0.8532	$\mathbf{341.2201 \pm 96.6132}$	$\mathbf{0.8639 \pm 0.1177}$

Table 5
Synthetic data performance of GenerativeMTD and *k*NNMTD for regression datasets.

Data	Method	*k*	PCD	KSTest	CSTest	TSTR	TRTS	TRTR	TSTS	DCR	NNDR
			(↓)	(↑)	(↑)	RMSE Score (↓)				(↑)	(↑)
Thyroid	*k*NNMTD	3	**0.2117**	**0.9314**	**0.9971**	**2.3698**	**3.5751**	**5.5509**	**3.3199**	0.1433 ± 0.3424	0.0640 ± 0.1714
	GenerativeMTD	6	0.3731	0.7329	0.9848	6.7934	6.6397	4.9925	3.5892	$\mathbf{33.4984 \pm 31.9310}$	$\mathbf{0.9002 \pm 0.1211}$
Liver	*k*NNMTD	3	**0.2354**	**0.9302**	–	**9.0375**	**11.0775**	**18.8987**	**11.1739**	0.2824 ± 0.4691	0.0241 ± 0.0461
	GenerativeMTD	8	0.5243	0.6617	–	21.3937	34.6077	20.1129	34.1588	$\mathbf{66.4342 \pm 50.6624}$	$\mathbf{0.9236 \pm 0.1080}$
Pima	*k*NNMTD	3	**0.4392**	**0.9516**	**0.9497**	2.8338	3.9195	6.8093	3.4750	0.0866 ± 0.3020	0.0048 ± 0.0172
	GenerativeMTD	9	0.7286	0.6310	0.9397	**0.3312**	**0.5121**	**0.3375**	**0.2515**	$\mathbf{57.8178 \pm 27.8601}$	$\mathbf{0.9285 \pm 0.0749}$
Prostate	*k*NNMTD	3	**0.4031**	**0.8854**	**0.9762**	**0.3896**	0.4720	0.8317	0.4955	0.1033 ± 0.2903	0.0365 ± 0.1172
	GenerativeMTD	10	0.6100	0.7262	0.8073	0.8189	**0.4771**	**0.7700**	**0.2928**	$\mathbf{11.2659 \pm 8.0597}$	$\mathbf{0.7964 \pm 0.1739}$
Fertility	*k*NNMTD	6	1.9965	**0.8800**	0.7114	**0.1143**	**0.1037**	**0.1797**	**0.1163**	0.0098 ± 0.0068	0.0323 ± 0.0644
	GenerativeMTD	3	**1.2685**	0.4350	**0.7216**	0.4431	0.3890	0.1796	0.2044	$\mathbf{1.1027 \pm 0.4084}$	$\mathbf{0.8380 \pm 0.1756}$
Bioconcentration	*k*NNMTD	3	0.6648	**0.9380**	**0.9866**	**0.3413**	**0.4249**	0.7689	0.4088	0.0026 ± 0.0037	0.0022 ± 0.0051
	GenerativeMTD	10	**0.6551**	0.7722	0.6272	0.8420	0.5262	**0.6305**	**0.3686**	$\mathbf{2.3936 \pm 0.8188}$	$\mathbf{0.9037 \pm 0.0968}$
Heart failure	*k*NNMTD	5	0.8846	**0.9217**	0.9490	**507.3344**	**580.1821**	**955.7350**	**548.6953**	103.2856 ± 221.1156	0.0509 ± 0.1088
	GenerativeMTD	5	**0.6829**	0.5639	**0.9619**	994.8289	1342.6452	1012.9494	586.8402	$\mathbf{1262.8066 \pm 972.5670}$	$\mathbf{0.7118 \pm 0.2584}$
Fat	*k*NNMTD	3	**0.6636**	**0.9358**	–	**0.6514**	**1.0991**	**1.6544**	**0.8967**	0.2646 ± 0.2740	0.0206 ± 0.0232
	GenerativeMTD	6	1.0328	0.7090	–	3.4595	3.2099	1.3504	2.7956	$\mathbf{47.3405 \pm 21.619}$	$\mathbf{0.9608 \pm 0.0420}$
Parkinson's	*k*NNMTD	4	**0.9996**	**0.9271**	**0.9468**	**0.0113**	**0.0156**	0.0272	**0.0120**	0.0823 ± 0.2085	0.0042 ± 0.0068
	GenerativeMTD	9	1.2171	0.8329	0.8221	0.0426	0.0219	**0.0224**	0.0169	$\mathbf{26.5523 \pm 14.9309}$	$\mathbf{0.8204 \pm 0.1703}$
Communities and Crimes	*k*NNMTD	4	18.0100	**0.9096**	0.0000	**0.0794**	**0.1128**	0.1937	**0.1040**	0.0146 ± 0.0072	0.0092 ± 0.0052
	GenerativeMTD	6	**16.8932**	0.5846	**0.3449**	0.1971	0.2927	**0.1654**	0.1675	$\mathbf{3.0106 \pm 0.2590}$	$\mathbf{0.9724 \pm 0.0217}$
Sweat Study (Case Study)	*k*NNMTD	3	**1.1659**	**0.9365**	**0.9782**	**0.3410**	**0.5075**	0.8332	**0.4218**	0.9001 ± 1.1498	0.0027 ± 0.0027
	GenerativeMTD	5	1.4521	0.7414	0.9564	0.6936	0.7847	**0.8296**	0.6012	$\mathbf{330.8251 \pm 92.3862}$	$\mathbf{0.8528 \pm 0.1316}$

Table 6
Running time of GenerativeMTD for 200 epochs.

Task	Data	Running time (min)
Classification	*Mammographic mass*	165.00
	Caesarean	76.00
	Cryotherapy	74.00
	Immunotherapy	75.00
	Post-operative	86.00
	Wisconsin breast cancer	139.00
	Cleveland heart disease	23.00
	Cervical	104.00
	Sweat Study	91.00
	Urban land cover	161.00

(*continued on next page*)

Table 6 (*continued*).

Task	Data	Running time (min)
Regression	*Thyroid*	76.00
	Liver	84.00
	Pima	173.00
	Prostate	98.00
	Fertility	88.00
	Bioconcentration	200.00
	Heart failure	97.00
	Fat	90.00
	Parkinson's	94.00
	Sweat Study (Regression)	93.00
	Communities and Crime	90.00

References

[1] L. Xu, M. Skoularidou, A. Cuesta-Infante, K. Veeramachaneni, Modeling tabular data using conditional gan, 2019, arXiv preprint arXiv:1907.00503.

[2] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, A.P. Sales, Generation and evaluation of synthetic patient data, BMC Med. Res. Methodol. 20 (1) (2020) 1–40, http://dx.doi.org/10.1186/s12874-020-00977-1.

[3] T. Che, Y. Li, A.P. Jacob, Y. Bengio, W. Li, Mode regularized generative adversarial networks, 2016, arXiv preprint arXiv:1612.02136.

[4] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, Y. Kim, Data synthesis based on generative adversarial networks, 2018, arXiv preprint arXiv:1806.03384.

[5] J. Jordon, J. Yoon, M. Van Der Schaar, PATE-GAN: Generating synthetic data with differential privacy guarantees, in: International Conference on Learning Representations, 2018.

[6] F.J. Moreno-Barea, J.M. Jerez, L. Franco, Improving classification accuracy using data augmentation on small data sets, Expert Syst. Appl. 161 (2020) 113696.

[7] J. Sivakumar, K. Ramamurthy, M. Radhakrishnan, D. Won, Synthetic sampling from small datasets: A modified mega-trend diffusion approach using k-nearest neighbors, Knowl.-Based Syst. (2021) 107687.

[8] J.T. Tomás, N. Spolaôr, E.A. Cherman, M.C. Monard, A framework to generate synthetic multi-label datasets, Electron. Notes Theor. Comput. Sci. 302 (2014) 155–176.

[9] J. Raymaekers, P. Rousseeuw, Cellwise: Analyzing data with cellwise outliers. R package version 2.2. 5, 2020.

[10] J.A. Sáez, Noise simulation in classification with the noisemodel R package: Applications analyzing the impact of errors with chemical data, J. Chemometr. 37 (5) (2023) e3472.

[11] S. Zhao, J. Song, S. Ermon, Infovae: Information maximizing variational autoencoders, 2017, arXiv preprint arXiv:1706.02262.

[12] C. Villani, Optimal Transport, Old and New. Notes for the 2005 Saint-Flour Summer School, in: Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer, 2008.

[13] I. Tolstikhin, O. Bousquet, S. Gelly, B. Schoelkopf, Wasserstein autoencoders, 2017, arXiv preprint arXiv:1711.01558.

[14] O. Bousquet, S. Gelly, I. Tolstikhin, C.-J. Simon-Gabriel, B. Schoelkopf, From optimal transport to generative modeling: the VEGAN cookbook, 2017, arXiv preprint arXiv:1705.07642.

[15] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, A. Smola, A kernel method for the two-sample-problem, Adv. Neural Inf. Process. Syst. 19 (2006) 513–520.

[16] Y. Li, K. Swersky, R. Zemel, Generative moment matching networks, in: International Conference on Machine Learning, PMLR, 2015, pp. 1718–1727.

[17] G.K. Dziugaite, D.M. Roy, Z. Ghahramani, Training generative neural networks via maximum mean discrepancy optimization, 2015, arXiv preprint arXiv:1505.03906.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Adv. Neural Inf. Process. Syst. 27 (2014).

[19] E. Choi, S. Biswal, B. Malin, J. Duke, W.F. Stewart, J. Sun, Generating multi-label discrete patient records using generative adversarial networks, in: Machine Learning for Healthcare Conference, PMLR, 2017, pp. 286–305.

[20] S. Patel, A. Kakadiya, M. Mehta, R. Derasari, R. Patel, R. Gandhi, Correlated discrete data generation using adversarial training, 2018, arXiv preprint arXiv:1804.00925.

[21] Z. Che, Y. Cheng, S. Zhai, Z. Sun, Y. Liu, Boosting deep learning risk prediction with generative adversarial networks for electronic health records, in: 2017 IEEE International Conference on Data Mining, (ICDM), IEEE, 2017, pp. 787–792.

[22] R. Venugopal, N. Shafqat, I. Venugopal, B.M.J. Tillbury, H.D. Stafford, A. Bourazeri, Privacy preserving generative adversarial networks to model electronic health records, Neural Netw. 153 (2022) 339–348.

[23] A. Srivastava, L. Valkov, C. Russell, M.U. Gutmann, C. Sutton, Veegan: Reducing mode collapse in gans using implicit variational learning, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 3310–3320.

[24] Z. Zhao, A. Kunar, H. Van der Scheer, R. Birke, L.Y. Chen, CTAB-GAN: Effective table data synthesizing, 2021, arXiv preprint arXiv:2102.08369.

[25] C. Ma, J. Li, M. Ding, B. Liu, K. Wei, J. Weng, H.V. Poor, RDP-GAN: A Rényi-differential privacy based generative adversarial network, 2020, arXiv preprint arXiv:2007.02056.

[26] S. Suh, H. Lee, P. Lukowicz, Y.O. Lee, CEGAN: Classification enhancement generative adversarial networks for unraveling data imbalance problems, Neural Netw. 133 (2021) 69–86.

[27] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, T. Aila, Training generative adversarial networks with limited data, Adv. Neural Inf. Process. Syst. 33 (2020) 12104–12114.

[28] A.B.L. Larsen, S.K. Sønderby, H. Larochelle, O. Winther, Autoencoding beyond pixels using a learned similarity metric, in: International Conference on Machine Learning, PMLR, 2016, pp. 1558–1566.

[29] Datacebo, The synthetic data vault. Put synthetic data to work!, 2023, [Accessed 22-Jul-2023].

[30] J. Jordon, J. Yoon, M. Van Der Schaar, Measuring the quality of synthetic data for use in competitions, 2018, arXiv preprint arXiv:1806.11345.

[31] M.N. Fekri, A.M. Ghosh, K. Grolinger, Generating energy data for machine learning with recurrent generative adversarial networks, Energies 13 (1) (2019) 130.

[32] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.

[33] K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1615–1630.

[34] A. Alemi, B. Poole, I. Fischer, J. Dillon, R.A. Saurous, K. Murphy, Fixing a broken ELBO, in: International Conference on Machine Learning, PMLR, 2018, pp. 159–168.

[35] R. Shu, H.H. Bui, S. Zhao, M.J. Kochenderfer, S. Ermon, Amortized inference regularization, Adv. Neural Inf. Process. Syst. 31 (2018).

[36] A. Radhakrishnan, K. Yang, M. Belkin, C. Uhler, Memorization in overparameterized autoencoders, 2018, arXiv preprint arXiv:1810.10333.

[37] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of wasserstein gans, 2017, arXiv preprint arXiv:1704.00028.

[38] J. Feydy, T. Séjourné, F.-X. Vialard, S.-i. Amari, A. Trouvé, G. Peyré, Interpolating between optimal transport and MMD using sinkhorn divergences, in: The 22nd International Conference on Artificial Intelligence and Statistics, PMLR, 2019, pp. 2681–2690.

[39] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, Adv. Neural Inf. Process. Syst. 26 (2013) 2292–2300.

[40] A. Genevay, G. Peyré, M. Cuturi, Learning generative models with sinkhorn divergences, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2018, pp. 1608–1617.

[41] C. Frogner, C. Zhang, H. Mobahi, M. Araya-Polo, T. Poggio, Learning with a wasserstein loss, 2015, arXiv preprint arXiv:1506.05439.

[42] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, Ann. Math. Stat. 35 (2) (1964) 876–879.

[43] [dataset], D. Dua, C. Graff, UCI machine learning repository, 2017, http://archive.ics.uci.edu/ml.

[44] [dataset], J.J. Faraway, Linear Models with R, CRC Press, 2014.

[45] [dataset], M. Elter, R. Schulz-Wendtland, T. Wittenberg, The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process, Med. Phys. 34 (2007) 4164–4172, http://dx.doi.org/10.1118/1.2786864.

[46] [dataset], M.Z. Amin, A. Ali, Performance evaluation of supervised machine learning classifiers for predicting healthcare operational decisions, Wavy AI Res. Found: Lahore, Pakistan (2018) 1–8.

[47] [dataset], F. Khozeimeh, R. Alizadehsani, M. Roshanzamir, A. Khosravi, P. Layegh, S. Nahavandi, An expert system for selecting wart treatment method, Comput. Biol. Med. 81 (2017) 167–175, http://dx.doi.org/10.1016/j.compbiomed.2017.01.001.

[48] [dataset], W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, 87, 1990, pp. 9193–9196, http://dx.doi.org/10.1073/pnas.87.23.9193,
[49] [dataset], W.H. Wolberg, O. Mangasarian, T.F. Coleman, Y. Li, Pattern recognition via linear programming: theory and application to medical diagnosis, in: Large-Scale Numerical Optimization, SIAM Publications, 1990, pp. 22–30.
[50] [dataset], K.P. Bennett, O.L. Mangasarian, Robust linear programming discrimination of two linearly inseparable sets, Optim. Methods Softw. 1 (1992) 23–34, http://dx.doi.org/10.1080/10556789208805504.
[51] [dataset], Sobar, R. Machmud, A. Wijaya, Behavior determinant based cervical cancer early detection with machine learning algorithm, Adv. Sci. Lett. 22 (2016) 3120–3123, http://dx.doi.org/10.1166/asl.2016.7980.
[52] [dataset], B. Johnson, Z. Xie, Classifying a high resolution image of an urban area using super-object information, ISPRS J. Photogramm. Remote Sens. 83 (2013) 40–49.
[53] [dataset], B.A. Johnson, High-resolution urban land-cover classification using a competitive multi-scale object-based approach, Remote Sens. Lett. 4 (2013) 131–140.
[54] [dataset], D. Gil, J.L. Girela, J. De Juan, M.J. Gomez-Torres, M. Johnsson, Predicting seminal quality with artificial intelligence methods, Expert Syst. Appl. 39 (2012) 12564–12573, http://dx.doi.org/10.1016/j.eswa.2012.05.028.
[55] [dataset], F. Grisoni, V. Consonni, S. Villa, M. Vighi, R. Todeschini, QSAR models for bioconcentration: is the increase in the complexity justified by more accurate predictions?, Chemosphere 127 (2015) 171–179, http://dx.doi.org/10.1016/j.chemosphere.2015.01.047.
[56] [dataset], F. Grisoni, V. Consonni, M. Vighi, S. Villa, R. Todeschini, Investigating the mechanisms of bioconcentration through QSAR classification trees, Environ. Int. 88 (2016) 198–205, http://dx.doi.org/10.1016/j.envint.2015.12.024.
[57] [dataset], D. Chicco, G. Jurman, Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone, BMC Med. Inform. Decis. Mak. 20 (2020) 16, http://dx.doi.org/10.1186/s12911-020-1023-5.
[58] [dataset], M.A. Little, P.E. McSharry, S.J. Roberts, D.A. Costello, I.M. Moroz, Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection, BioMed. Eng. Online 6 (2007) 1, http://dx.doi.org/10.1186/1475-925X-6-23.
[59] [dataset], M. Redmond, A. Baveja, A data-driven software tool for enabling cooperative information sharing among police departments, Eur. J. Oper. Res. 141 (2002) 660–678.
[60] [dataset], P. Pearlmutter, G. DeRose, C. Samson, N. Linehan, Y. Cen, L. Begdache, D. Won, A. Koh, Sweat and saliva cortisol response to stress and nutrition factors, Sci. Rep. 10 (2020) 1–11, http://dx.doi.org/10.1038/s41598-020-75871-3.
[61] S. van Buuren, Multiple imputation of discrete and continuous data by fully conditional specification, Stat. Methods Med. Res. 16 (3) (2007) 219–242, http://dx.doi.org/10.1177/0962280206074463.
[62] S. van Buuren, K. Groothuis-Oudshoorn, Mice: Multivariate imputation by chained equations in R, J. Stat. Softw. 45 (3) (2011) 1–67, http://dx.doi.org/10.18637/jss.v045.i03.