



FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation:

Computational Engineering

The spring semester, 2022

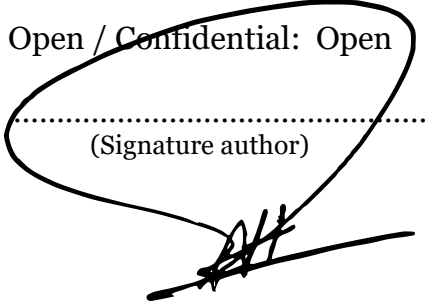
Author: Alireza Hossein Zadeh Nik

Open / Confidential: Open

Program coordinator: Aksel Hiorth

Supervisor(s): Michael Alexander Riegler
Andrea Storås

.....
(Signature author)



Thesis title: The Generation of Synthetic Healthcare Data
Using Deep Neural Networks

Credits (ECTS): 30

Keywords:

Synthetic data generation, Tabular data,
Healthcare, Generative Adversarial Network,
Privacy

Pages: 79

+ appendix: 36

Stavanger, June 25, 2022



University of
Stavanger

**Faculty of Science and Technology
Department of Energy Resources (IER)**

The Generation of Synthetic Healthcare Data Using Deep Neural Networks

Master's Thesis in Computational Engineering

by

Alireza Hossein Zadeh Nik

Supervisors:

Michael Alexander Riegler

Andrea Storås

Program coordinator:

Aksel Hiorth

June 25, 2022

Abstract

High-quality tabular data is a crucial requirement for developing data-driven applications, especially healthcare-related ones, because most of the data nowadays collected in this context is in tabular form. However, strict data protection laws introduced in Health Insurance Portability and Accountability (HIPAA) and General Data Protection Regulation (GDPR) present many obstacles to accessing and doing scientific research on healthcare datasets to protect patients' privacy and confidentiality. Thus, synthetic data has become an ideal alternative for data scientists and healthcare professionals to circumvent such hurdles. Although many healthcare data providers still use the classical de-identification and anonymization techniques for generating synthetic data, deep learning-based generative models such as Generative Adversarial Networks (GANs) have shown a remarkable performance in generating tabular datasets with complex structures. Thus, this thesis examines the GANs' potential and applicability within the healthcare industry, which often faces serious challenges with insufficient training data and patient records sensitivity.

We investigate several state-of-the-art GAN-based models proposed for tabular synthetic data generation. Precisely, we assess the performance of TGAN, CTGAN, CTABGAN, and WGAN-GP models on healthcare datasets with different sizes, numbers of variables, column data types, feature distributions, and inter-variable correlations. Moreover, a comprehensive evaluation framework is defined to evaluate the quality of the synthetic records and the viability of each model in preserving the patients' privacy. After training the selected models and generating synthetic datasets, we evaluate the strengths and weaknesses of each model based on the statistical similarity metrics, machine learning-based evaluation scores, and distance-based privacy metrics.

The results indicate that the proposed models can generate datasets that maintain the statistical characteristics, model compatibility, and privacy of the original ones. Moreover, synthetic tabular healthcare datasets can be a viable option in many data-driven applications. However, there is still room for further improvements in designing a perfect architecture for generating synthetic tabular data.

Acknowledgments

First of all, I am deeply grateful to the Simula Metropolitan Center for introducing me to the topic of generative modeling and allowing me to use their state-of-the-art hardware in my research. I would like to give a special thanks to my head supervisor Michael Alexander Riegler for his generous support and invaluable guidance during the thesis. I am also grateful to Andrea Storås for her valuable feedback and editing help during the whole process.

Additionally, I would like to express my gratitude to my program coordinator, Aksel Hiorth, for his immense support throughout my M.Sc. degree in Computational Engineering at the Department of Energy Resources (IER) at the University of Stavanger.

Lastly, I would be remiss in not mentioning my family and friends, especially my parents, without whom this incredible journey would not have been possible.

Contents

Abstract	i
Acknowledgements	ii
1 Introduction	1
1.1 Motivation	1
1.2 Problem Definition	2
1.2.1 Objectives	3
1.2.2 Research Methodology Overview	3
1.3 Ethical Considerations	4
1.4 Main contributions	5
1.5 Thesis Outline	5
2 Background & Preliminaries	7
2.1 Artificial Neural Network	7
2.1.1 Feed Forward Neural Network	8
2.1.2 Stochastic Optimization	9
2.1.3 Batch Normalization	10
2.2 Generative Adversarial Network	11
2.2.1 GAN Framework	11
2.2.2 GAN Training	12
2.2.3 GAN Failure Modes	14
2.3 Different Types of GANs	16
2.3.1 Deep Convolutional GAN (DCGAN)	17
2.3.2 Conditional GAN (cGAN)	17
2.3.3 Wasserstein GAN (WGAN)	18
2.3.4 WGAN-GP	21
2.3.5 GANs and Tabular Data Generation	22
2.4 Related Works	23

3	Method	27
3.1	GAN-based Models for Tabular SDG	27
3.1.1	TGAN	27
3.1.2	CTGAN	30
3.1.3	TableGAN	33
3.1.4	CTABGAN	35
3.1.5	WGAN	37
3.2	Evaluation Framework	38
3.2.1	Statistical Resemblance (General Utilities)	38
3.2.2	Specific Utilities (Machine Learning Utilities)	42
3.2.3	Preserved Privacy	43
4	Experimental Setup	45
4.1	Data	45
4.1.1	Epileptic Seizure Recognition	46
4.1.2	Diabetes	47
4.1.3	Thyroid Disease	48
4.1.4	MIMIC III	48
4.2	Implementation Details	51
5	Evaluation & Results	53
5.1	Statistical Resemblance	53
5.1.1	Basic Statistical Check	53
5.1.2	Column-wise Associations	55
5.1.3	Column Distributions	57
5.2	Machine Learning-based Evaluation	59
5.3	Preserved Privacy	61
5.4	Summary of the Overall Results	63
6	Discussion	64
6.1	In-depth analysis of Distributions	64
6.1.1	Multi-modal Numerical Columns	64
6.1.2	Discrete Numerical Columns	65
6.1.3	Long-tailed Numerical Columns	66
6.1.4	Imbalanced Categorical Columns	67
6.2	Limitations	69
6.2.1	Data	69
6.2.2	Hyper-parameter Selection	69
6.2.3	Convergence	70
7	Conclusions & Recommendations	71

7.1	Conclusions	71
7.2	Future Work	73
Bibliography		74
A	Diabetes	80
A.1	Basic Similarities	80
A.2	Pair-wise Correlation	81
A.3	Column Distributions	82
	A.3.1 Continuous Columns	82
	A.3.2 Categorical Columns	85
A.4	General Utility Scores	90
A.5	ML Cross-Testing Scores	90
B	MIMIC III	91
B.1	Basic Similarities	91
B.2	Pair-wise Correlation	92
B.3	Column Distributions	93
	B.3.1 Continuous Columns	93
	B.3.2 Categorical Columns	95
B.4	General Utility Scores	97
B.5	ML Cross-Testing Scores	97
C	Thyroid	98
C.1	Basic Similarities	98
C.2	Pair-wise Correlation	99
C.3	Column Distributions	100
	C.3.1 Continuous Columns	100
	C.3.2 Categorical Columns	102
C.4	General Utility Scores	108
C.5	ML Cross-Testing Scores	108
D	Epileptic	109
D.1	Basic Similarities	109
D.2	Pair-wise Correlation	110
D.3	Column Distributions	111
	D.3.1 Continuous Columns	111
D.4	General Utility Scores	115
D.5	ML Cross-Testing Scores	115

List of Abbreviations:

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the Curve
CGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
CS	Chi-squared test
CTGAN	Conditional Tabular Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
DCR	Distance to Closest Record
DL	Deep Learning
EHR	Electronic Health Record
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
GP	Gradient Penalty
GPU	Graphics Processing Unit
HIPAA	Health Insurance Portability and Accountability
JS	Jensen–Shannon divergence
KL	Kullback–Leibler divergence
KS	Kolmogorov–Smirnov test
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
ReLU	Rectified Linear Activation function
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics curve
SDG	Synthetic Data Generation
TGAN	Tabular Generative Adversarial Network
UiS	University of Stavanger
WGAN	Wasserstein Generative Adversarial Neural Network

Chapter 1

Introduction

1.1 Motivation

Today, Artificial Intelligence (AI) is a game-changer that offers promising opportunities in many application domains such as the transportation and automotive industry, banking, social media, gaming, etc. However, this impact has been undeniably slower in the medical applications [20] due to the unavailability and imbalanced nature of the patients' Electronic Health Records (EHRs) to the broader AI research community. A significant reason for this is the Privacy Protection concerns and the extremely time-consuming approval process for accessing the data. The Health Insurance Portability and Accountability Act (HIPAA) in the United States and the General Data Protection Regulation (GDPR) in European countries introduce restrictive regulations against the research and analysis of electronic healthcare records to protect patients' privacy [76]. Such laws would severely slow the pace of data analysis and research in the health care industry. EHR providers typically protect the patients' privacy through de-identification and anonymization techniques, including removing identifiable or quasi-identifiable features, perturbing them, or grouping variables into higher-level categories to remove sensitive data [76]. However, there is no guarantee that the anonymized patient data would not be used to re-identify patients and their sensitive information. Several studies have shown that when anonymized data is linked to other publicly available datasets (social media, etc.), the peoples' identities and their sensitive information may be re-identified [16].

Alternatively, Synthetic Data Generation (SDG) unveils interesting properties to circumvent the legal restrictions of healthcare datasets and is an ideal replacement for the data anonymized by classical de-identification techniques. The main point of SDG is to synthesize new data through automated processes that preserve the

underlying structure and statistical properties of the original sensitive data to prevent people’s privacy from being compromised. Synthetic health data generation can help medical practitioners to share data without any privacy violation and use the synthetic data as an addition to the health data itself [54]. To achieve an appropriate generalization, Deep Learning models and Machine Learning algorithms need a large amount of training data. Thus, a reliable synthetic dataset can be used for augmentation tasks and training robust models that are typically not trainable with the available data.

Generally, synthetic data generation techniques are classified into two groups [20]. The *Process-driven* techniques synthesize data by a pre-determined mathematical model of an underlying process like Monte Carlo simulation. These methods are heavily dependent on external information and domain-specific knowledge. On the other hand, the *Data-driven* methods synthesize data by fitting a joint multi-variate probability distribution on the given dataset and then sampling from it [74]. Example models include Gaussian Mixture models, Bayesian networks, and Gaussian Copulas. Nevertheless, most of these generative models have restrictions related to the model complexity, dataset size, and available probability distributions.

In recent years, deep learning generative models have delivered outstanding performance in representing complex distribution functions, flexibility, and fidelity of the synthetic data. Several deep generative architectures have been proposed for the synthetic data generation (SDG) tasks, including Variational Auto-Encoders, Deep Belief Nets, and Generative Adversarial Networks (GANs). Lately, GANs have gained overwhelming popularity for their successful applications within image and video generation domains. However, a survey conducted by the *Kaggle* platform showed that structured data is the most common format in industry and the second most common in academic environments [64, 74]. This tendency to tabular data format has motivated researchers to conduct studies to apply GAN’s strategy for synthesizing this specific format, although most of the GAN’s studies have been focused on unstructured data like images. Several studies have recently explored different variants of GANs to see whether this adversarial architecture is a better choice than its statistical counterparts [50, 74, 76].

1.2 Problem Definition

In this thesis, we take advantage of GAN’s adversarial strategy for training a privacy-preserving synthetic data generator to synthesize real tabular healthcare datasets. We aim to investigate multiple GAN variants proposed for the tabular data generation in the medical domain. While there exist several papers proposing a GAN variant for the synthetic data generation tasks, each of them utilizes different datasets

and evaluation metrics [20]. Moreover, the selected evaluation setup should capture the inter-variable dependencies and marginal distributions simultaneously and reflect the preserved privacy between the generated and the real samples. Thus, the evaluation framework must include metrics in different dimensions.

On the other hand, tabular datasets have other issues regarding having various data types that require different encoding for GAN’s applications. These challenges make a direct comparison of SDG based on GAN’s adversarial properties extremely difficult. Consequently, this thesis aims to develop guidelines for comparing and evaluating different tabular GAN methods in generating synthetic electronic health-care records. All in all, this leads to the research objectives outlined below.

1.2.1 Objectives

- Finding a publicly available healthcare tabular dataset rich enough for the synthetic data generation task. The source of the data and the data itself should be thoroughly studied to grasp the meaning and characteristics of constructing features.
- Selecting a suitable evaluation framework to assess the quality of the synthesized data and reliability of the model in preserving the privacy of the patient records.
- Conducting a systematic study of multiple tabular GAN models for generating synthetic patient health records and comparing them based on the different metrics of our selected evaluation framework.

1.2.2 Research Methodology Overview

Several tabular GAN-based models are experimented on multiple healthcare datasets to assess the potential of synthetic data in the healthcare domain. *Diabetes* and *Thyroid Disease* datasets from UCI Machine Learning Repository [13], *Epileptic Seizure Recognition* dataset from Kaggle and MIMIC III (Multi-parameter Intelligent Monitoring in Intensive Care) clinical database [33] are selected for our synthetic data generation tasks. We choose these datasets because they are the most comprehensive medical yet publicly available datasets containing tens of thousands of records with different data types and statistical distributions. Each proposed GAN model is trained on a clean, pre-processed version of the selected datasets. Then, for each case study, we generate a synthetic tables of the same size as the original one and evaluate it based on the proposed evaluation framework. We use novel evaluation

metrics gathered from multiple publications to capture *Statistical Resemblance*, *Machine Learning utility* and *Preserved Privacy*. The privacy metric is needed to ensure generated samples are different enough from training records. This evaluation setup can assess whether any form of data analysis on the original dataset would yield similar results as it is done on the synthetic one. Figure 1.1 shows an overview of the proposed methodology.

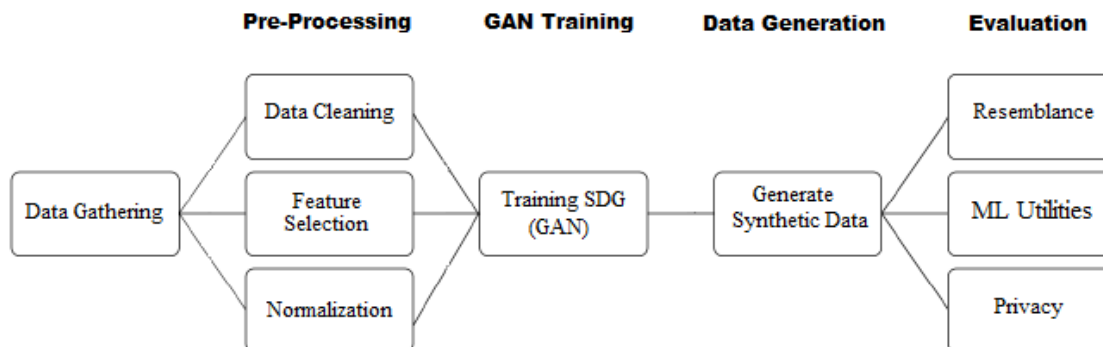


Figure 1.1: An overview of the proposed research methodology

1.3 Ethical Considerations

With the rise of ML and AI technologies in peoples' lives on a daily basis, new ethical challenges have emerged that endanger civil liberties. These legitimate concerns should be addressed in the policies and regulations defining how the AI systems should be evolved, how they should be applied in our daily lives and how they affect the society. One of these ethical concerns, especially when using AI technologies in the healthcare domain, is privacy [9, 17]. To circumvent the privacy barrier, synthetic data generation is a viable alternative since synthetic records capture the statistical properties and inter-variable relationships of the original data. At the same time, they do not reveal the patients' sensitive information. Although the generative models deliver outstanding performance in synthetic data generation, the sensitive information in the original data is still susceptible to being leaked in the generated data, and the patients' privacy is likely to be compromised in the synthetic data generation process. Thus, the researchers often train synthetic generative models using a differentially private mechanism [1, 14] to generate synthetic records in a privacy-preserving manner and ensure people's confidentiality.

1.4 Main contributions

The main contributions of the thesis are listed as the following:

- We carry out a comprehensive survey on existing medical tabular data sets and their applicability to be used for generating synthetic data.
- We implement a complete pre-processing pipeline for each healthcare tabular dataset to make them ready and feasible for the synthetic data generation task.
- We perform detailed experiments comparing different tabular generative adversarial networks and their configurations.
- We conduct a survey on existing metrics and their capabilities to be used for bench-marking synthetic medical data.
- We develop an evaluation framework and perform a systematic study on the quality of synthetic medical data using different metrics.
- The main findings and contributions of this work are summarized in a research article that will be submitted to the journal of *Artificial Intelligence for Medicine*.

1.5 Thesis Outline

In Chapter 1, we introduce the motivation for generating synthetic health data and provide an overview of the research methodology. The rest of the thesis is structured as follows:

- Chapter 2 presents the theoretical background, including an introduction to Neural Networks, the concept of Generative Adversarial Networks (GAN), distinguished GAN architectures, GAN's failure modes, their difficulties in generating tabular data, and the related works in the tabular SDG field.
- Chapter 3 details the proposed tabular generation models. Five tabular GAN-based models are selected to be tested in the healthcare domain. Finally, we describe the evaluation framework used in the thesis.
- Chapter 4 provides a complete description of the implementation details and the healthcare datasets selected for our experiments.
- In Chapter 5, we evaluate the results of the conducted experiments and compare each tabular SDG technique using the various evaluation metrics.

- In chapter 6, we conduct an in-depth analysis of the properties of the investigated SDG models and introduce their limitations
- Finally, we conclude the thesis in Chapter 7 with our findings and provide possible suggestions for future studies.

Chapter 2

Background & Preliminaries

This chapter presents the theoretical background of the thesis. First, we discuss several concepts related to training artificial neural networks. Then, we introduce the Generative Adversarial Network (GAN), GAN's failure modes, distinguished GAN architectures, and their difficulties in generating tabular data. Finally, we present a detailed discussion regarding related works in the tabular synthetic data generation field.

2.1 Artificial Neural Network

Artificial Neural Networks (ANNs), commonly known as Neural Networks, are a subset of Artificial Intelligence (AI) and are the basis of Deep Learning (DL) algorithms. The name and structure of ANNs are hugely inspired by how the biological neurons inside the human brain function. As the biological neurons receive a signal from one neuron and transmit it to another, each artificial neuron (node) gets an input, produces the output, and passes it as input to another node. A neural network is a collection of artificial neurons interconnected to one another through edges aiming to learn a non-linear mathematical representation that best maps a set of inputs to a group of outputs. The connecting edges have weights that are updated during the training process. The output of each neuron is calculated by applying an activation function to the weighted sum of all the neuron's inputs. These activations are usually non-linear functions helping the network to learn the non-linearity of the features of the information. Due to their tremendous capacity, ANNs can solve various Machine Learning and Deep Learning tasks in different domains [7, 18, 21, 43].

2.1.1 Feed Forward Neural Network

Feed-Forward Neural Network is the most straightforward version of the ANNs as the data only moves in one direction in the network. It is a set of interconnected, sequential layers consisting of an input layer, one or more hidden layers, and an output layer [18, 21, 27]. A simple feed-forward neural network is illustrated on the left side of figure 2.1. The input layer contains three nodes representing the number of features in the input, while each hidden layer includes four artificial neurons. Lastly, the network's output is computed in the last layer consisting of two neurons. Since every node in each layer is connected to all the nodes in the previous and next layers, the layers are referred to as fully connected ones.

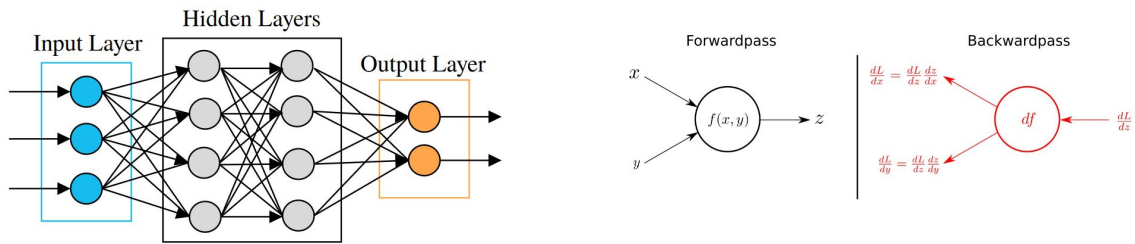


Figure 2.1: An illustration of a fully-connected neural network on the left side [49]. The right side of the figure shows an example of the forward and backward propagation in a neural network [38].

The training of a feed-forward neural network is conducted in three steps. First, the input goes through a series of transformations to produce the network's output. Specifically, for each neuron in each hidden layer, we apply an activation function to the weighted sum of all neuron's inputs from the previous layer to calculate each node's output. Starting from the input layer and iteratively repeating the same calculations for each neuron in the subsequent layers would eventually result in the network's output. This step is called forward propagation (forward pass) [18, 21].

The second step involves updating the network's parameters according to how wrong the network's output is compared to the desired one. After each forward propagation step, a loss function is calculated between the network's output and the expected one. Then, we calculate the gradient of the loss function with respect to each network's parameter using the chain rules [18, 21]. An example of the forward and backward propagations is illustrated on the right side of figure 2.1.

After each backward pass, we update the network's parameters to minimize the loss function. Gradient-based optimization algorithms like Stochastic Gradient Descent make this parameter adjustment. The gradient descent algorithm enables the

network to find the minimum value of its loss function by iteratively updating the parameters in the opposite direction of the gradient of the loss. We repeat the forward pass, backward pass, and parameter adjustments until the network does not show any improvement by continuing the training process [18, 21].

2.1.2 Stochastic Optimization

As discussed in the previous section, the core of training an ANN is an optimization problem. The stochastic optimization algorithms are responsible for updating the network’s parameters to enhance the learning process. In other words, these optimizers seek to maximize or minimize the network’s loss function with respect to its parameters.

Stochastic Gradient Descent (SGD):

Gradient descent is a gradient-based optimization algorithm that aims to find a global minimum for the loss function with respect to the network’s parameters. After each backpropagation step, the gradient descent adjusts the network’s parameters for the next iteration in the opposite direction of the gradient of the loss function [21, 43, 58]. This parameter (θ) update is conducted based on the following formula:

$$\theta_{n+1} = \theta_n - \gamma \nabla_{\theta_n} \mathbb{L}(x, \theta_n)$$

γ denotes the learning rate and is used to tune the step size in which the optimization algorithm moves toward the global minimum of the network’s loss function \mathbb{L} . Although stochastic gradient descent is one of the most efficient and effective approaches for optimizing many Machine Learning models, it can be a significantly slow algorithm for complex neural architectures.

Adam:

Adam is one of the most popular algorithms for stochastic optimization. It was introduced by Kingma and Ba in their 2015 ICLR paper [37], and the algorithm’s name originated from adaptive moment estimation. Adam is the recommended optimization algorithm for many deep learning-based architectures instead of classical stochastic gradient descent. They are computationally efficient, easy to implement, and have little memory requirement. While the gradient descent uses a single learning rate for updating the network’s parameters in the training process, Adam computes adaptive learning rates for each parameter individually from estimates of the first-order and second-order moments of the gradients. This optimizer calculates an exponential moving average of the gradient and the squared gradient, while

the hyper-parameters β_1 and β_2 control the decay rates of these moving averages. The pseudo-code below is taken from the original paper [37] showing the parameter update at step.

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t .

Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0 : Initial parameter vector
 $m_0 \leftarrow 0$ (Initialize 1st moment vector)
 $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
 $t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
 $t \leftarrow t + 1$
 $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
 $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
 $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
 $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
 $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
 $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

2.1.3 Batch Normalization

Batch Normalization is a novel algorithm for training faster and more stable ANNs introduced by Ioffe et al.[31] in 2015. They proposed normalizing the neurons' outputs in the hidden layers by using the mean and variance of each batch of the input data. This forces the outputs in each layer to follow a standard normal distribution across the current batch. This normalization technique first computes the μ and σ^2 as the mean and variance of the activation values across a batch of input data in each hidden layer. Then, it normalizes the activation outputs of each neuron in the hidden layer using formula 1.

$$(1) \quad Z_{norm}^{(i)} = \frac{Z^{(i)} - \mu}{\sqrt{\sigma^2 - \epsilon}} \qquad (2) \quad \hat{Z} = \gamma * Z_{norm}^{(i)} + \beta$$

Finally, to give the neural network flexibility for selecting the optimal distribution for each hidden layer's outputs, the algorithm applies a linear transformation with two trainable parameters as formula 2. Specifically, the parameters γ and β are used to

calibrate the standard deviation and the bias (shifting) of the output distributions, respectively, and they are trained through gradient descent optimization.

The experiments have shown that the Batch Normalization significantly improves the speed and stability of training neural networks and allows using higher learning rates in the training process without any adverse effect on the convergence.

2.2 Generative Adversarial Network

Generative Adversarial Network (GAN) is one of the most prominent unsupervised model architectures in generative modeling. The architecture was first introduced by Ian Goodfellow et al.[22] as part of the advances in the Neural Information Processing Systems (NIPS) conference in 2014. Goodfellow proposed a novel generative architecture that automatically discovers the patterns in the original data and generates new realistic fake samples indistinguishable from the original ones. Since then, a plethora of scientific publications have focused on different aspects of GANs, and many deep learning generative models have been adopted based on this architecture. Specifically, GANs have delivered outstanding performances in image and video generation in the past few years, and it is still one of the most active research topics in the field of unsupervised generative modeling.

2.2.1 GAN Framework

The GAN architecture consists of two networks: a generator and a discriminator [21, 22]. The generator learns the implicit patterns and distributions of the original data and generates realistic fake records that possibly could have been drawn from the same distribution as the original data. On the other hand, the discriminator estimates the probability of a sample coming from the original or synthetic data, classifying records into real or fake. The basic concept of GAN is adopted from game theory, where the generator and discriminator compete against each other in a min-max game. This two-player competition aims to train these networks simultaneously until reaching the Nash Equilibrium. At this ideal point, the generator creates realistic replicas utterly indistinguishable from the original data, and the discriminator cannot classify between the real and fake records, guessing randomly with the probability of 0.5. In simple words, the generator and discriminator can be thought of as an art forger and a museum inspector. The forger tries to replicate the original paintings so he can sell them to the museum, while the inspector struggles to detect the forged paintings from the original ones. The forger and inspector improve their capabilities simultaneously until the forger can create realistic paintings undetectable to the inspector. Figure 2.2 illustrates the components of the GAN framework.

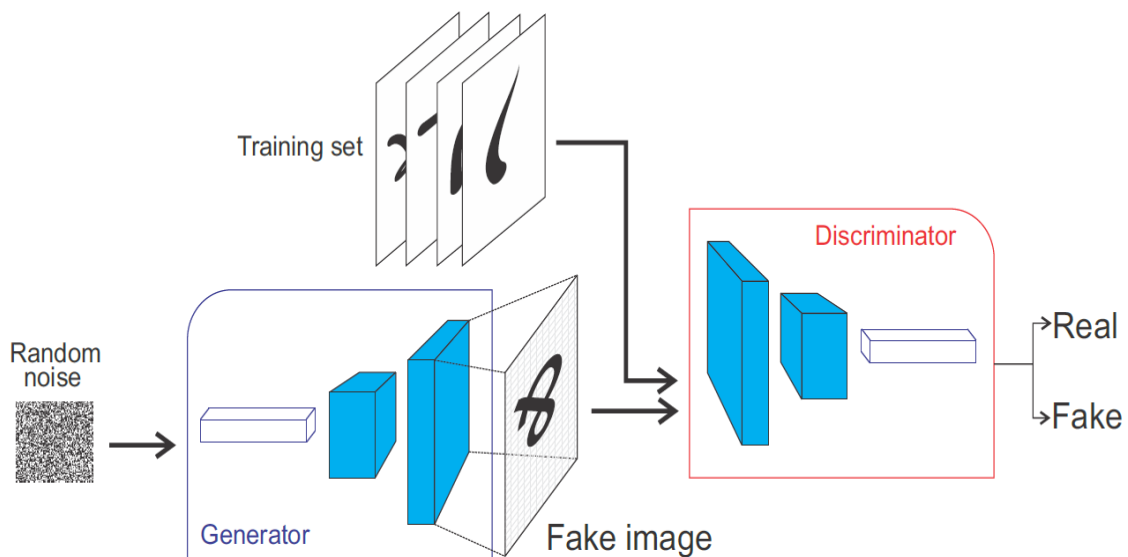


Figure 2.2: Generative Adversarial Network (GANs) framework [55].

2.2.2 GAN Training

The generator (G) and discriminator (D) networks in the GAN architecture are two differentiable functions with respect to their parameters. The generator's input is a random vector sampled from a prior noise distribution $\mathbb{P}_z(z)$. Although the original paper used uniform distribution for sampling noise vector, the normal distribution has become an ideal choice recently. The generator's goal is to learn the latent space of the original data and then sample from this latent space to generate new realistic fake records. To achieve this goal, the generator translates the noise vector into the points in the problem domain and optimizes its weights (θ_g) when playing a min-max game with the discriminator. The discriminator takes input from the original or synthetic (output of the generator) examples and outputs a single value representing the probability of the discriminator's inputs belonging to the real or fake ones. In other words, the discriminator's goal is to assign correct labels and optimizes the weights (θ_d) when playing a min-max game with the generator, while the generator's goal is to complicate the discriminator's job by generating realistic fake data [21, 22]. Figure 2.3 depicts a schematic of training a typical GAN.

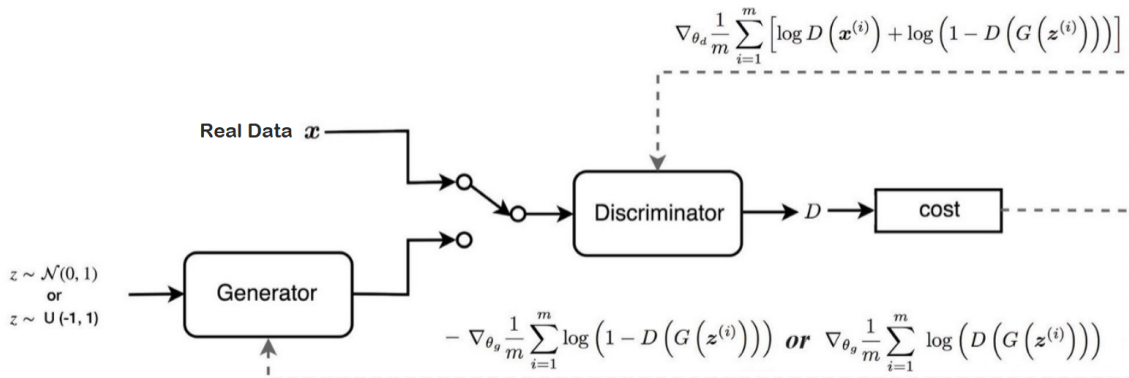


Figure 2.3: A schematic of GAN training [30].

The training algorithm involves iterative gradient optimizations for both adversarial components [21, 30]. In each training iteration, we optimize the discriminator and generator networks consecutively. First, a minibatch of random noises is sampled to generate a minibatch of fake records, followed by sampling a minibatch of records from the original training records. Then, we use the original and generated records to conduct a gradient optimization step on the discriminator to reduce its loss function (L_d) and update its weights (θ_d). After the discriminator optimization, we perform an optimization step on the generator network. We sample a minibatch of random noises as input to the generator and use the backpropagation to update the generator's weights (θ_g) and reduce its loss function (L_g). In both the generator and discriminator scenarios, we can only control the parameters of the network we are optimizing through backpropagation. In other words, although each adversarial component's objective function depends on the other component, each one is entitled to modify its own weights in the optimization steps. Ideally, these gradient-based optimizations continue until reaching an equilibrium where both loss functions of the generator and discriminator are at a local minimum with respect to their weights (Nash Equilibrium). To capture the opposing interactions between the generator and discriminator, the authors of the original paper [22] proposed using the overall min-max value function mathematically defined as:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))]$$

The above overall min-max formula can be separated into the value functions for generator and discriminator:

$$\begin{aligned} \max_D V(D) &= \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))] \\ \min_G V(G) &= \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z)))] \end{aligned}$$

As observed, the discriminator’s objective is to maximize the probability of identifying the correct labels. In other words, it tries to maximize the probability of assigning the original examples with label one and the generated (fake) records with label zero. On the other hand, the generator seeks to minimize the chance of the synthetic records being identified as fake by the discriminator. However, the generator cannot improve in this min-max game in practice. At the beginning of the training process, the generator creates random noises that the discriminator can easily classify as fake records. This results in the generator losing its ability to improve as quickly as the discriminator due to the saturation of the generator’s loss. To circumvent this problem, Goodfellow et al. suggested using a non-saturating version of the original min-max loss function [21, 22]. They proposed a subtle modification to the min-max generator loss while keeping the discriminator loss unchanged. The modified generator loss is defined as:

$$\max_G V(G) = \mathbb{E}_{x \sim p_z} [\log(D(G(z)))]$$

Rather than training the generator to minimize the probability of the synthetic records being identified as the fake ones by the discriminator, the authors proposed flipping the labels of the generated records (labels 1 one instead of 0) and training the generator to maximize the probability of the generated records being classified as original ones. In practice, the outcome of this modified loss function does not alter. However, it results in much stronger gradients and fixes the generator’s saturation problem to a great extent [30].

2.2.3 GAN Failure Modes

GANs are powerful tools yet difficult and tricky to train. There are several scenarios in which GAN architectures are doomed to failure. The most frequent GAN failures are non-convergence and mode-collapse.

Non-convergence (Instability):

The GANs are challenging to train due to the simultaneous training of the adversarial components. The generator and discriminator compete against each other in a min-max (zero-sum) game during the training process until they reach an equilibrium [6]. In each optimization step, the improvement to one component comes at the cost of degradation of the other one. Although these two cost functions may converge, there are many use cases in which the gradient descent fails to find a local equilibrium in the non-convex, min-max games. In other words, the gradient descent is not necessarily a proper choice to train a stable GAN having a min-max loss function. To identify the non-convergence issue, the discriminator’s loss dramatically

drops to zero, while the generator’s loss may increase significantly or drops to zero in the same period. This failure results in the generated records having an extremely poor quality that the discriminator can easily classify. Researchers are constantly proposing new cost functions to mitigate the non-convergence issue. However, there is no solid evidence that these proposed cost functions would always yield stable performance [6, 29].

Mode Collapse:

Mode collapse is one of the most encountered and well-recognized failure scenarios in training GANs. It is referred to as a phenomenon in which the generator only learns certain regions (modes) of the original data distribution. In other words, the generator generates samples with remarkably low diversity. The generator learns to produce new fake records from one or few modes of the original data distribution and misses the rest [6, 29]. Figure 2.4 illustrates the effects of the mode collapse phenomenon. Comparing the target image with the generated images in different training epochs, we observe that the Vanilla GAN (top row) faces mode collapse and its generator only learns a few modes from the target distribution, while the Reg-GAN [10] architecture succeeds in capturing all the modes and generates samples statistically and visually similar to the target distribution.

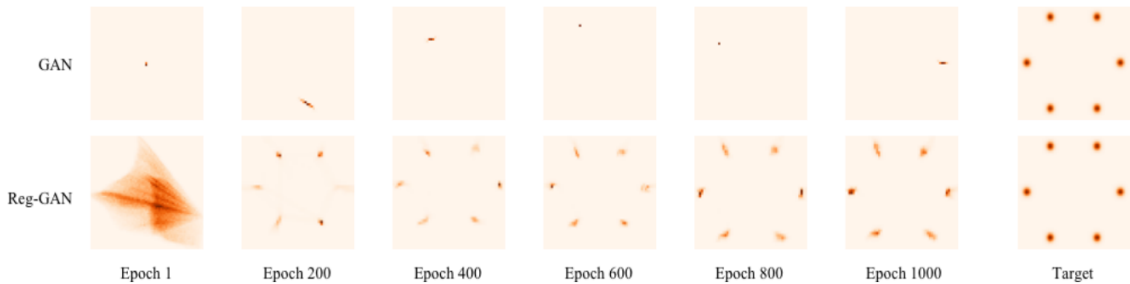


Figure 2.4: Generated samples of two GANs trained on a 2D Gaussian dataset [10].

There is no clear consensus on why precisely the mode collapse scenario happens. Several papers have attempted to address the causes of this failure mode. We believe that using ill-suited loss functions is a significant cause of mode collapse in training GANs. To elaborate on the effects of the loss function, consider $P(x)$ and $Q(x)$ as the original and synthetic data distributions, respectively. Theoretically, the GAN’s objective is to minimize the distance between the $P(x)$ and $Q(x)$ distributions, and the ideal case happens when $Q(x) = P(x)$ [48]. In probability theory, we often use statistical divergences like Jensen-Shannon (JS) divergence and Kullback-Leibler (KL) divergence to measure the distance between two probability distributions [57].

In other words, the GAN seeks to minimize the divergence of $P(x)$ and $Q(x)$. The GAN divergence and JS divergence are remarkably similar, with some slight differences. Poole et al.[52] demonstrated that the GAN divergence (JS divergence) tends to fit a limited number of modes perfectly (mode-seeking) instead of capturing all the modes of the original distribution (mode-covering). This results in the generator producing new records with less diversity (mode collapse). Figure 2.5 illustrates the mode-seeking and mode-covering approximations of an original bimodal distribution.

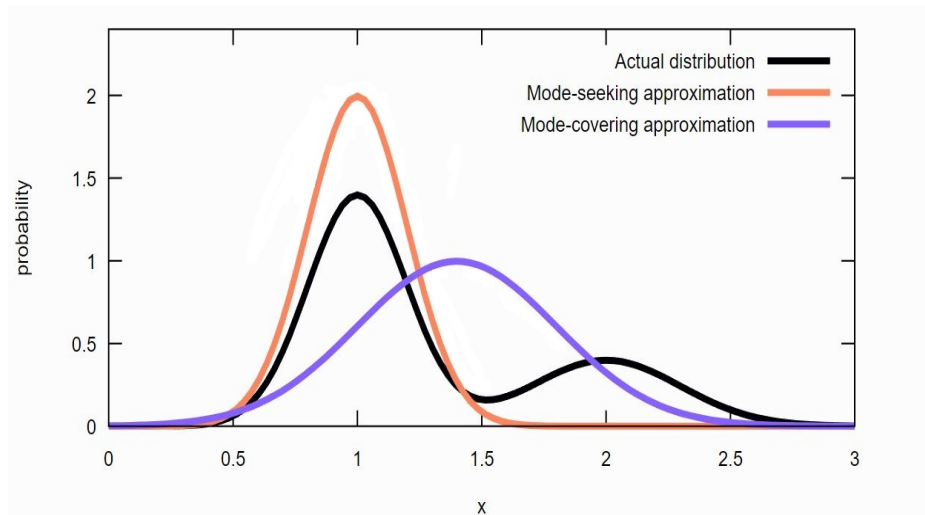


Figure 2.5: An illustration of mode-seeking and mode-covering approximations [48].

Although different tricks and architectures have been proposed to mitigate the mode collapse issue, it is yet to be solved completely in training GANs. For instance, several papers suggested using mini-batch discrimination [59], feature matching [59], unrolled GANs [44], Wasserstein distance instead of the original loss function [3], etc., to tackle mode collapse.

2.3 Different Types of GANs

Since the introduction of GANs in 2014, many research papers have adopted this novel framework and proposed minor or significant modifications to the original architecture. This results in the creation of hundreds of GAN-based models in different areas. This section introduces a few of these architectures that provided the basis for more advanced models.

2.3.1 Deep Convolutional GAN (DCGAN)

Perhaps one of the GAN’s early innovations that laid the foundation for developing many other models is the Deep Convolutional GAN (DCGAN). The architecture was first introduced by Radford et al in [53]. They proposed using the de-convolutional and convolutional layers in the discriminator and generator, respectively. The DCGAN is one of the most recommended architectures, especially when developing GAN-based models for image generation tasks. The authors made specific recommendations for training a stable GAN based on the deep convolutional neural network architecture. They proposed eliminating the fully connected layers and replacing the pooling layers with strided convolutions and deconvolutions. The generator network comprises deconvolutional layers for upsampling, batch normalization, and ReLU activation function for all layers except for the output layer (tanh). On the other hand, the discriminator network consists of convolutional layers for downsampling, batch normalization, and LeakyReLU for all internal layers except for the last layer, which uses the sigmoid activation function. Figure 2.6 illustrates the generator and discriminator of the DCGAN architecture in detail.

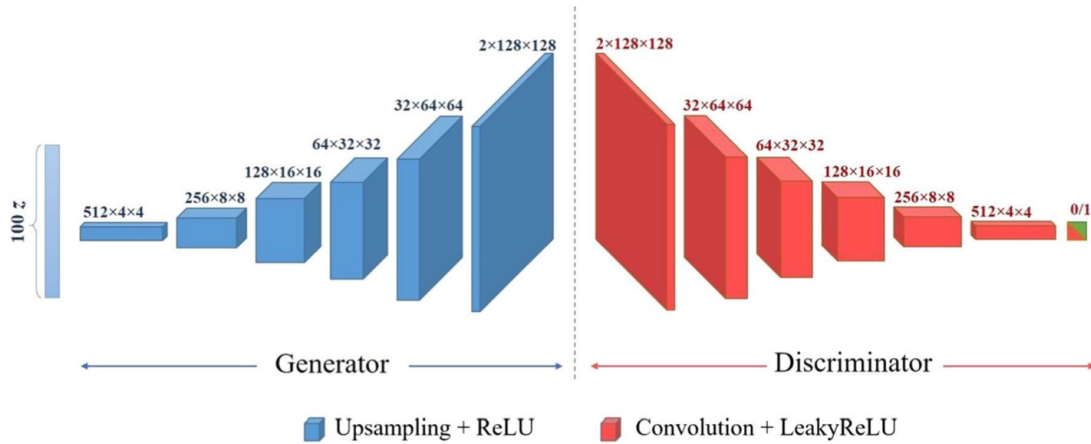


Figure 2.6: The Deep Convolutional GAN (DCGAN) architecture [35]

2.3.2 Conditional GAN (cGAN)

In all GAN architectures discussed so far, synthetic data generation occurs randomly from the latent space input to the points in a specific domain. In other words, we do not have any control over the types of the generated records. However, it is possible to use additional information in the generator and discriminator to improve the training process and conduct a targeted data generation. This type of architecture is called Conditional GAN (cGAN), and it was first introduced by Mirza et al. [46] in 2014. They proposed extending the original GAN architecture by adding auxiliary

information (as class labels y) to the generator and discriminator as conditions to control the synthetic data generation process. The cGAN’s modified objective function can be defined as [46]:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}} [\log D(x|y)] + \mathbb{E}_{z \sim p_z} [\log(1 - D(G(z|y)))]$$

To demonstrate the robustness of the cGAN training process, they concatenated the one-hot encoded class labels of the MNIST handwritten digit dataset [40] with the inputs of each adversarial network. The process is illustrated in figure 2.7.

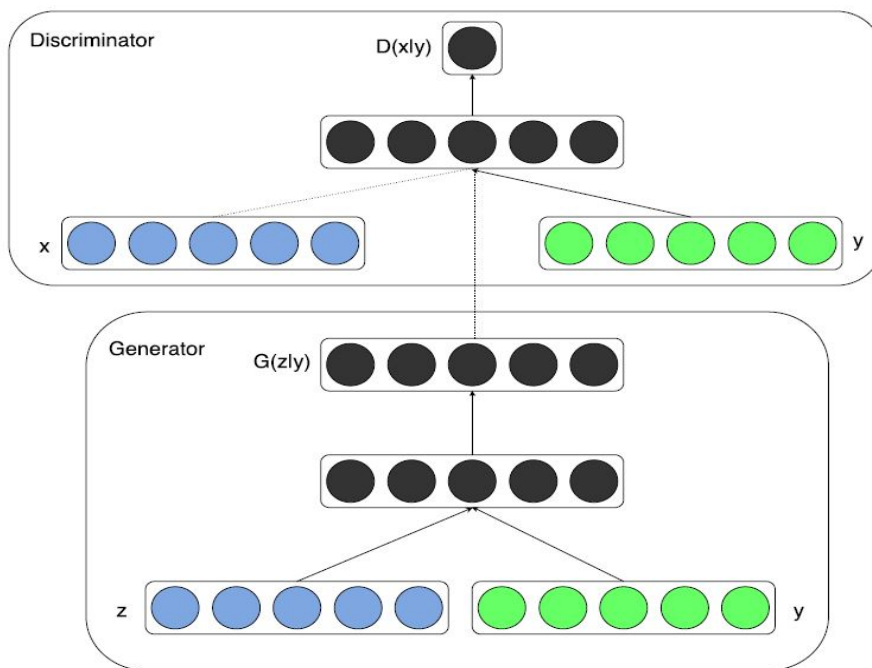


Figure 2.7: Adversarial components of the cGAN model [46].

2.3.3 Wasserstein GAN (WGAN)

In 2017, Arjovsky et al.[3] proposed another state-of-the-art adaptation of the original GAN framework. They introduced Wasserstein GAN (WGAN) to overcome the problems associated with the loss functions presented in the original GAN paper. They showed that if the model is trained using the original min-max loss function, the generator cannot improve when the discriminator significantly outperforms it, especially at the beginning of the training process. This happens due to the vanishing gradients of the JS divergence between the original and generated data distributions

when both distributions are quite distant, which results in the saturation of the generator’s loss. Furthermore, they demonstrated that the non-saturating version of the original loss function proposed to fix the vanishing gradient issues makes the model more unstable. This instability is due to the great variance of the gradients in the modified loss function.

To circumvent the mentioned issues, Arjovsky et al.[3] proposed an alternative loss function based on the Wasserstein distance between the original and synthetic data distributions. The Wasserstein (Earth-mover) metric compares two probability distributions and measures their distance as the minimum cost (effort) required to transform one distribution into another. Intuitively, suppose the synthetic distribution is assumed to be a pile of soil. In that case, the Earth-mover metric measures the minimum cost associated with moving and transforming the pile of soil into the location and the shape of the original distributions respectively. This distance metric is mathematically defined as:

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

$\Pi(P_r, P_g)$ represents all probable transportation plans for the synthetic data distribution to transform into the original one, while γ denotes a specific plan. The Earth-mover distance is continuous and differentiable everywhere, and in contrast to the JS divergence, it has no upper limit for its gradient. In other words, the gradient of the Wasserstein metric continues to grow regardless of the distance between the original and synthetic distributions, making the model less susceptible to vanishing gradients [6, 28]. Thus, Arjovsky et al. proposed a loss function based on the Wasserstein metric to produce smoother gradients in the training process. Their proposed loss function measures the Earth-mover distance between the original and synthetic distributions. However, instead of a discriminator outputting probability to classify the original and synthetic records, the WGAN model uses a neural network outputting a scalar value that scores how real or fake the given inputs are. Since this network in the WGAN architecture does not have a sigmoid function to conduct the classification tasks, it is called *Critic* to reflect the difference from the discriminator [3, 6, 28]. The rest of the WGAN network is the same as the original GAN architecture, as observed in figure 2.8.

However, it is improbable to approximate the Earth-mover distance in the WGAN framework based on the Wasserstein mathematical expression because the formula seeks to find the minimum cost from an infinitive number of joint distributions (transportation plans) [6, 28]. Hence, the authors in [3] proposed using Kantorovich-Rubinstein Duality to simplify the calculations as:

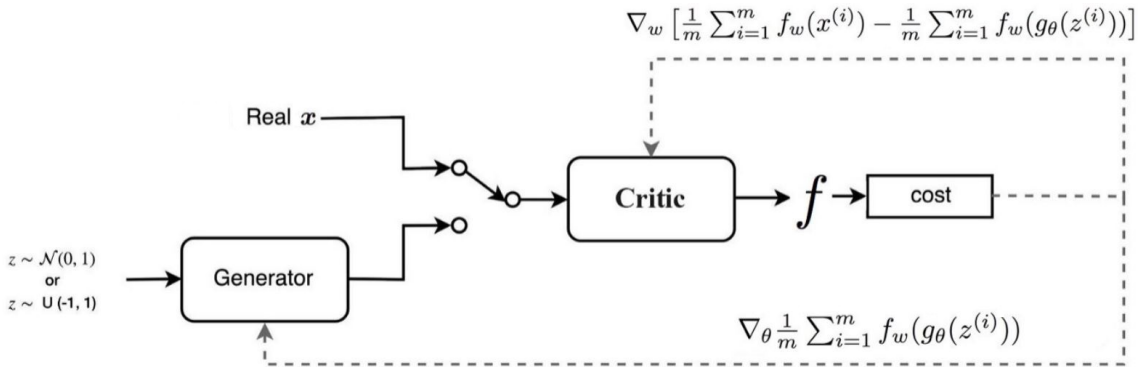


Figure 2.8: A schematic of WGAN training [28]

$$W(P_r, P_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_r} [f(x)] - \mathbb{E}_{x \sim p_g} [f(x)]$$

In this formulation, f is a differentiable function whose gradients have a norm of at most 1 for every point. In other words, the slope of this continuous function can only range between -1 and 1. This function (f) is a 1-Lipschitz (1-L) continuous, and the particular condition is called the 1-L continuity constraint [6, 28]. Arjovsky et al.[3] demonstrated that to approximate the Wasserstein distance in the WGAN framework and limit the loss function from growing too much; the critic network should be 1-L. To enforce the 1-L continuity constraint on the critic network, the original paper proposed the weight clipping technique to enforce the critic's parameters range in a fixed interval. When training the critic network, this interval is controlled by a clipping hyper-parameter. After each gradient optimization, if the critic's weights are outside of a desired range of values, they are clipped to assure that weights are updated in that specific interval. However, as noted by Arjovsky et al.[3], the weight clipping technique is a sub-optimal strategy that limits the learning ability of the critic. Especially if the clipping hyper-parameter is incorrectly tuned, it may lead to vanishing gradients or non-convergence issues. In other words, the model performance is significantly prone to the changes in the clipping hyper-parameter. To circumvent these issues, a promising alternative for enforcing the 1-L continuity constraint on the critic network is described in Section 2.3.4 in detail.

In the WGAN framework, the critic is trained to learn the best fitting 1-L continuous function (f) to help approximate the Wasserstein distance while optimizing the loss function. The WGAN loss function is described mathematically in [3] as:

$$L(P_r, P_g) = W(P_r, P_g) = \max_{\omega \in \mathcal{W}} \mathbb{E}_{x \sim p_r} [f_\omega(x)] - \mathbb{E}_{z \sim p_g(z)} [f_\omega(G_\theta(z))]$$

The critic aims to maximize the Earth-mover distance between the original and synthetic data samples (maximizing the critic loss). At the same time, the generator tries to minimize the distance (minimizing the generator loss). As the generator and critic improve in the training process, the critic finds a best-fitting function to the original data when the Earth-mover distance gets smaller, and the generator produces synthetic records that resemble the original data distribution [3, 6, 28]. Moreover, the authors proposed updating the critic k times per generator optimization to ensure that the generator network is improved in the proper direction. In summary, the WGAN model has two significant benefits compared to other GAN architectures:

- The model is significantly less prone to mode collapse in the training process.
- The generator can still improve itself when the original and synthetic distributions are far apart (no vanishing gradients).

2.3.4 WGAN-GP

WGAN-GP is a modified version of the Wasserstein GAN architecture proposed by Gulrajani et al.[23] in 2017. They suggested using an improved technique to enforce the 1-L continuous constraint on the critic network instead of using the weight clipping strategy. As noted in the original WGAN paper [3], the model performance is susceptible to the clipping hyper-parameter if the WGAN model uses the weight clipping approach. In other words, an incorrect choice of this hyper-parameter may lead to vanishing or exploding gradients during the training process. To circumvent these issues, Gulrajani et al.[23] proposed a gradient penalty technique to penalize the gradient norm of the critic network if it deviated from 1 (definition of 1-L continuity) and named the modified model WGAN-GP. The loss function of the critic network in the WGAN-GP is defined as:

$$L_d = \mathbb{E}_{x \sim p_r} [f_w(x)] - \mathbb{E}_{z \sim p_g(z)} [f_w(G_\theta(z))] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]$$

The last term of the formula corresponds to the gradient penalty, and the \hat{x} represents uniformly straight lines pairs of points sampled from the original and generated data distributions. The authors recommended setting the λ hyper-parameter to 10 and avoiding the batch normalization technique in the critic’s network. Their experiments showed that batch normalization might decrease the effectiveness of the gradient penalty [23, 28].

WGAN with gradient penalty remarkably improves training stability and convergence, and it is an effective model in many applications. However, the gradient penalty causes an undesirable computational complexity to the training process.

2.3.5 GANs and Tabular Data Generation

The GAN architectures elaborated in the previous sections are commonly applied in image generation tasks. However, many real-world applications rely on tabular data consisting of columns of various data types. While, for generating fake images, we train GANs to learn the distributions of pixel values ranging from 0 to 255; in tabular datasets, GANs are trained to find a joint distribution of variables of different data types such as: continuous (floating), categorical, integers and Boolean. Consequently, producing synthetic tables raises extra challenges compared to generating fake images due to the nature of tabular data. This section addresses some of the significant challenges of designing a tabular GAN architecture [36, 75].

Different Data Transformations:

As with every other deep learning-based model, the data should be represented appropriately to be applicable for training GANs. A tabular dataset with different data types requires different data transformation techniques. In image generation tasks, we simply use a min-max scaler to transform pixel values into the range of -1 to 1. However, in many real-world tabular datasets, the values in the continuous columns have non-Gaussian, complex distributions as multi-modal or long-tailed ones [79]. In these cases, using a simple data transformation technique like the min-max scaling limit the GAN model from learning the complex numerical distributions and lead to vanishing gradients. Furthermore, the categorical columns should apply different transformation techniques to find appropriate data representations for training GANs compared to numerical columns. Thus, the pre-processing step in each tabular GAN architecture must include various kinds of data transformations suited for specific column distributions, and the data types for all columns should be correctly specified for the GAN model [5, 75]. Otherwise, it treats all data types in a similar manner. For instance, if we do not differentiate between a one-hot encoded label column and a continuous one, the GAN model generates a floating feature when replicating the label column.

Categorical Data Generation:

Generating categorical values in GAN architectures is not as straightforward as synthesizing continuous ones. For continuous variables, we normalize the input values to $[-1, 1]$ and use the tanh activation function on the generator's output layer to produce values in the same range [5]. A typical approach to producing categorical values is using the Softmax function on the network's last layer, outputting the probabilities corresponding to each label, and then selecting the most likely one using the argmax function. However, generating categorical values is not as easy as using a Softmax activation function followed by an argmax operation. The argmax is

not a differentiable function and can not be used for backpropagation in the training process. An easy workaround to circumvent this issue could be omitting the argmax and only using the Softmax activation function outputting probabilities. However, in this situation, the discriminator network easily differentiates between the Softmax probabilities and the one-hot encoded ground truth [5, 42]. This would prevent both the generator and discriminator improve in proper directions.

Generally, there are two approaches to circumvent this issue [5, 42]. First, we can add noise to the one-hot encoded original labels to make them similar to the Softmax outputs. Thus, the discriminator cannot easily differentiate between the Softmax probabilities and one-hot encoded ground truth labels. The other approach is using Gumbel-Softmax to sample from categorical distributions in a differentiable way. This technique was introduced by Jang et al. in [32] proposing a novel method to reparameterize the categorical distributions. Specifically, the authors leveraged Softmax activation with *temperature* to approximate a differentiable version of the argmax function. For more details, please refer to the original paper.

The GAN models presented so far would fail to generate categorical columns or continuous variables with complex distributions. More importantly, they fail to capture inter-variable dependencies between the features. Thus, many research papers have addressed all or some of these issues simultaneously to design a tabular synthesizer. The following section reviews the outstanding works in the tabular SDG.

2.4 Related Works

The topic of tabular synthetic data generation (SDG) has become an active research area for the scientific community in recent years. The SDG techniques are generally classified into process-driven and data-driven methods [20]. While the process-driven techniques synthesize data by a pre-determined computational model of an underlying process, the data-driven methods generate synthetic data by fitting a joint multi-variate probability distribution to the original data and then sampling from it. There are a plethora of publications regarding the use of process-driven and classical data-driven methods in tabular synthetic data generation. However, our work explicitly focuses on the deep learning-based tabular generative models. Specifically, we investigate the properties of the GAN-based models in generating tabular healthcare data. To reach our research objectives, we conduct a thorough literature study chronologically on the proposed GAN-based models tailored for tabular SDG.

MedGAN is one of the first GAN-based models, introduced in 2017 to generate discrete aggregated healthcare patient records. To circumvent the problems of training categorical columns in GAN architectures, Choi et al.[11] proposed using a pre-

trained auto-encoder to translate discrete values into continuous latent codes. The pre-trained decoder is placed between the generator and discriminator in the proposed architecture. While the generator tries to learn the continuous latent codes, the decoder translates the generator’s output to the original data format and passes it to the discriminator. Therefore, the discriminator is either fed with the original or fake records. There are a few functional limitations regarding the proposed SDG model. The MedGAN architecture can only generate discrete numerical and categorical features and does not support synthesizing more data types.

Although MedGAN architecture treats multi-categorical columns as a flat collection of binary variables, the model struggles to perform equally compared to the case when generating binary categorical columns. This is due to disregarding the data structure in the training process. To circumvent this, Camino et al., in their paper [8] proposed a sophisticated method to generate multi-categorical columns with GAN architectures. They modified the MedGAN architecture by splitting the output of the decoder with a dense layer for each categorical feature followed by a Gumbel-Softmax and then concatenating the results of the parallel layers to achieve the outcome. This modification increases the performance of the original MedGAN in generating multi-categorical features. However, it imposes limitations for knowing some additional information like the dimensionality of the features.

In another interesting work done by Park et al.[50], the TableGAN model was proposed to generate tabular datasets containing numerical and categorical columns with complex distributions. The adversarial components in this general-purpose synthesizer were adopted based on Deep Convolutional Neural Networks (DCNN) to capture inter-variable dependencies between columns. In addition to the generator and discriminator, the authors proposed an auxiliary classifier to increase the semantic integrity of the generated samples. Moreover, two additional loss terms were added to the generator’s loss to improve the training process. It is worth mentioning that the model treats categorical columns numerically after translating them to integers.

Lei Xu et al., in their scientific papers [74, 75], proposed TGAN and CTGAN tabular SDG models to address the shortcomings of the previous models. Both models use a mode-specific normalization technique to deal with the complexity of generating multi-modal numerical columns. While the authors used an LSTM network in the TGAN’s generator to synthesize each feature sequentially in the generation process, the CTGAN model uses a novel conditional generator and a unique training-by-sampling technique to deal with imbalanced categorical columns. Lastly, the TGAN and CTGAN models are adopted based on vanilla GAN and WGAN-GP architectures, respectively.

Baowaly et al.[4] proposed two modifications to the MedGAN architecture to improve the generation of discrete synthetic EHRs. They suggested using Wasserstein GAN and boundary-seeking GAN instead of the vanilla GAN architecture in the MedGAN model while keeping the remainder of both models' structures the same as MedGAN. The authors called the modified models MedWGAN and MedBGAN, respectively. Their analysis indicated that in all statistical similarity and machine learning-based evaluation metrics, the modified models outperform the MedGAN model, and the modifications improve training stability and model convergence. Specifically, the MedBGAN was the best-performing one among all three models.

CorGAN is another sophisticated modification of the MedGAN model proposed by Torfi et al.[65] to generate discrete and continuous numerical healthcare records. Like MedGAN's architecture, CorGAN uses a pre-trained auto-encoder to translate discrete values into continuous latent space. However, the authors used a convolutional GAN and a convolutional auto-encoder instead of the Multi-layer Perceptron (MedGAN) to capture the inter-variable dependencies effectively. Specifically, the generator, discriminator, and the auto-encoder's components use a 1-D convolutional structure, while the rest of the model is unaltered. Moreover, the proposed model demonstrated an acceptable degree of preserved privacy against Membership Inference attacks.

Recently, the research community has focused on protecting the SDG models against malicious attacks compromising the privacy and integrity of the sensitive information in the original training data. Several research papers, such as [34, 66, 67, 73] suggest using differentially private GAN-based architectures to provide privacy guarantees in the generation process. However, in complex use cases, it has been demonstrated that the quality of the synthetic records would decrease significantly in terms of statistical similarity and ML-based utilities when the noise is added in the generation process to ensure the differential privacy constraints. To circumvent these issues, Yoon et al.[78] suggested using a quantifiable definition for patients' identifiability instead of differential privacy constraints. Their proposed model, ADS-GAN, uses a modified conditional GAN framework while minimizing while assuring a specific identifiability level is met. The authors showed that the ADS-GAN model outperforms other differentially private models in all evaluation metrics by conditioning on an optimized set of variables instead of a pre-determined one.

Lastly, CTABGAN [79] is one of the most recent GAN-based models developed in the realm of synthetic data generation. Zhao et al.[79] adopted the core features of CTGAN and TableGAN models to handle the highly imbalanced categorical features and to improve generating skewed multi-modal and long-tailed continuous columns.

Although a plethora of GAN-based architectures is proposed for tabular healthcare generation tasks, most are designed for specific medical applications. For instance, many papers investigating the SDG in the healthcare domain use the MIMIC III clinical database to exclusively synthesize patients' ICD-9 codes (diagnostic codes). However, we intend to study the strengths and weaknesses of the SDG models in the healthcare domain that are not application-specific. In other words, we will investigate the GAN-based models capable of generating tabular healthcare datasets containing various data types and applicable to most medical applications. Thus, from all the SDG models introduced in this section, we only investigate the ones that are according to our objective and describe them thoroughly in Chapter 3.

Chapter 3

Method

In this chapter, we discuss two main topics. First, we describe the cutting-edge GAN-based models for tabular synthetic data generation. Five promising GAN-based SDG methods are summarized in detail. In the second section, we present the evaluation framework comprehensively and discuss the pros and cons of each metric individually.

3.1 GAN-based Models for Tabular SDG

3.1.1 TGAN

Tabular GAN(TGAN) is one of the first GAN-based architectures introduced to tackle the complexity of the generation of a synthetic table containing various data types. The model was first proposed by Lei Xu et al.[74] in 2018 with the goal of providing a general-purpose synthetic data generator that could synthesize the continuous and discrete (multinomial) columns of any tabular dataset simultaneously.

The authors proposed to use the Long Short-term Memory (LSTM) network with attention as the generator to synthesize columns in a sequential manner, while the attention mechanism is used to capture the inter-variable dependencies between features. Due to this mechanism, when a new column is to be generated sequentially, the dependencies of the previous, highly correlated columns can be captured effectively by the newly generated one. The discriminator, on the other hand, is just a Multi-layer Perceptron (a fully connected neural network). Figure 3.1 illustrates the generator and the discriminator separately and depicts how the TGAN is used to generate new tabular records.

Furthermore, the authors proposed a novel pre-processing step to circumvent the complexities regarding the continuous columns with arbitrary distributions (multi-modal). They realized that in many practical use cases, using the min-max normalization technique to center the non-Gaussian distributed values over $(-1,1)$ with a hyperbolic tangent activation function does not yield a proper result when synthesizing a multi-modal continuous distribution.

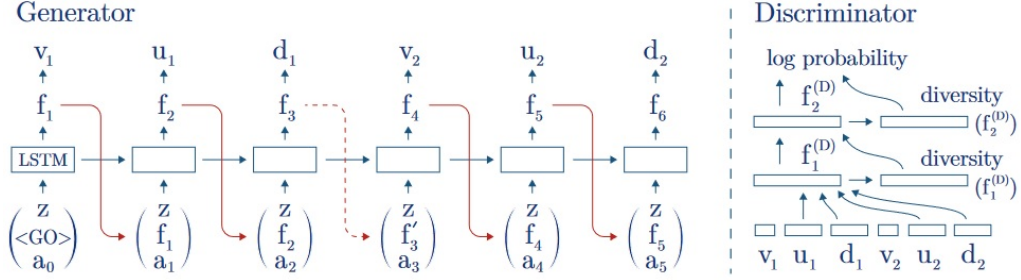


Figure 3.1: An illustration of the TGAN components [74].

Instead, they proposed a set of reversible transformations for the numeric features and called it the *mode-specific normalization*. In this approach, they train Gaussian Mixture model (GMM) on each individual continuous column to cluster the numeric values between different modes. More specifically, this method is performed based on the following steps [74]:

- First, a Gaussian mixture model is trained for each continuous feature C_i . The learned distribution for a column is defined as the weighted sum of k Gaussian components with $\eta_i^{(1)}, \dots, \eta_i^{(k)}$ and $\sigma_i^{(1)}, \dots, \sigma_i^{(k)}$ as means and standard deviation respectively.
- The normalized probabilities over k Gaussian components are calculated in form of a vector for each individual value $c_{i,j}$ in the column. Thus, for each value, we know the probabilities by which they belong to different modes. This normalized probability vector can be shown as $u_{i,j}^{(1)}, \dots, u_{i,j}^{(k)}$
- For each value $c_{i,j}$ in the column, we select the mode it most likely belongs to (among k components) and normalize the value based on the mean and standard deviation of the selected mode. This normalization is according to the $v_{i,j} = (c_{i,j} - \eta_i^{(m)}) / 2\sigma_i^{(m)}$, where $m = \text{argmax}_k u_{i,j}^{(k)}$. Lastly, the values are clipped to $[-0.99, 0.99]$.
- Finally, each continuous value $c_{i,j}$ is represented by $u_{i,j}$ and $v_{i,j}$.

The TGAN implementation uses a GMM with 5 modes ($k_{default} = 5$) and clusters all values in the numerical columns whether they are multi-modal or not. The TGAN

authors argue that if a continuous feature has a smaller number of modes (k_{actual}) compared to the default value of 5, the Gaussian mixture model assigns very low probabilities to the $5 - k_{actual}$ modes, and the learned distribution would only have k_{actual} modes.

Although many related works utilize Gumbel-Softmax [32] to make their model differentiable when generating categorical features, the TGAN implementation uses the Softmax activation function to generate the distribution of the categorical columns represented as one-hot encoded vectors. To eliminate the existing gap between the outputs of a Softmax function and the one-hot encoded representation of the real data, the model adds uniform noise to these binary features and re-normalizes them. Hence, they become indistinguishable to the discriminator. After the pre-processing step, each record of the original table is represented as below, where \oplus is the concatenation operator and $d_{i,j}$ is the one-hot encoded form of the categorical features:

$$r_j = v_{1,j} \oplus u_{1,j} \oplus \dots \oplus v_{N_c,j} \oplus u_{N_c,j} \oplus d_{1,j} \oplus \dots \oplus d_{N_d,j}$$

In the left side of the figure 3.1, we observe that the LSTM network (as the generator) generates 2 numerical and 2 categorical variables sequentially in 6 steps based on the actual order in the real data. Each numerical variable is generated in two steps including the generation of the value and the mode vector. On the other hand, the categorical features are generated in one step.

The right side of the figure 3.1 illustrates the TGAN discriminator as a fully connected neural network (MLP). The authors proposed to use the LeakyReLU activation function and Batch Normalization in all layers, while the Mini-batch Discrimination technique was used in the internal ones. Thus the output value of each layer in the discriminator network is computed based on [74]:

$$\begin{aligned} f_1^{(D)} &= \text{LeakyReLU}(\text{BN}(W_1^{(D)}(v_{1:n_c} \oplus u_{1:n_c} \oplus d_{1:n_d}))) \\ f_i^{(D)} &= \text{LeakyReLU}(\text{BN}(W_i^{(D)}(f_{i-1}^{(D)} \oplus \text{diversity}(f_{i-1}^{(D)})))) \end{aligned}$$

It should be mentioned that in the first layer, the numerical and categorical components generated from the LSTM cell are concatenated, as we concatenate the normalized features at the end of the pre-processing stage.

Regarding the choice of the loss function, the TGAN architecture uses the vanilla GAN losses to learn the marginal distribution of the individual columns. The only difference is that the authors add a KL divergence term to the generator loss to make the model more stable.

Based on the original paper [74], the generator loss can be defined as:

$$L_G = -\mathbb{E}_{z \sim N(0,1)} \log D(G(z)) + \sum_{i=1}^{n_c} \text{KL}(u'_i, u_i) + \sum_{i=1}^{n_d} \text{KL}(d'_i, d_i)$$

As observed, they modified the generator loss function of the vanilla GAN by adding the KL divergence of the categorical columns and the cluster vector of the numerical ones. Finally, it should be noted that the model uses ADAM as the optimizer.

3.1.2 CTGAN

The TGAN authors introduced another GAN-based tabular SDG model in 2019 to circumvent the challenges and issues associated with their former architecture [74]. As an example, the TGAN-generated samples usually face severe mode-collapse and lack the proper diversity, especially in the generated categorical columns. This issue is closely related to the imbalanced nature of many categorical features in real-world datasets, and it occurs because the discriminator finds it hard to detect whether the overall distribution includes the minor category or not. Hence, Xu et al.[75] introduced the Conditional Tabular GAN(CTGAN) to synthesize tabular data.

Although both models were introduced by the same authors, their architectures and implementations differ significantly. The only similarity both models share is the normalization step for continuous variables. As in the former model, CTGAN uses a mode-specific normalization technique to deal with the numerical features containing multi-modal distributions. One small upgrade is that the TGAN tries to fit a Gaussian Mixture Model (GMM) with m modes (the default value is 5) to cluster values of each continuous column, while CTGAN uses a Variational Gaussian Mixture Model(VGMM) to automatically calculate the number of modes. Moreover, instead of a normalized probability vector indicating the cluster each continuous value likely belongs to, the mode indicator in CTGAN implementation is represented as a one-hot vector. The CTGAN also uses a different approach to convert categorical columns into one-hot encoded vectors. Instead of adding noise to the one-hot encoded representation of the discrete features and normalizing them, the CTGAN implements Gumbel-Softmax [32].

To address the challenges associated with the class imbalance of categorical columns and to improve the training mechanism, the CTGAN paper proposed the use of a *Conditional Generator* and the *Training-by-Sampling* technique. The conditional generator allows for the generation of new samples conditioned on a specific value of a specific categorical column, thus all existing classes in the categorical features, whether majority or minority, are evenly sampled.

To modify the traditional GAN architecture into a conditional one, first, they intro-

duced the conditional vector *cond* to represent explicit conditioning in the generator and discriminator (critic). The *cond* vector is defined as $\mathbf{m}_1 \oplus \dots \oplus \mathbf{m}_{N_d}$, where \oplus is the concatenation operator and N_d is the total number of discrete features. This conditional vector is the concatenation of binary mask vectors of all categorical columns. Precisely, all these mask vectors are initially filled with zeros, then we choose the specific mask associated with the selected column in the condition and replace those elements having the same class as in the condition with 1. For instance, given two categorical columns $d_1 = \{1, 3, 5\}$ and $d_2 = \{2, 4\}$ with the condition ($d_1 = 1$), the mask vectors are represented as $\mathbf{m}_1 = [1, 0, 0]$ and $\mathbf{m}_2 = [0, 0]$ and consequently, the condition vector is $\text{cond} = [1, 0, 0, 0, 0]$.

The second challenge was to enforce the generator to synthesize samples that maintain the conditions they were given. By adding a penalty to the generator loss function that measures the cross-entropy between the conditional vector and generated binary representation of the categorical features, the model gets penalized if it deviates from the condition it was given.

Despite the proposition of the *cond* vector and conditional generator, there is no guarantee that all values in the categorical variables are evenly explored while the model is training. In other words, there is no guarantee that all classes (both the majority and minority classes) in the discrete columns are properly sampled in the training phase. To impose this condition, the CTGAN model uses a novel approach called Training-by-Sampling. In this technique, first, one of the discrete variables is randomly selected with equal probability. Then, the probability mass function (PMF) of the values within the selected feature is computed such that the probabilities are the logarithm of the frequency of each value. Thus, it is guaranteed that each category is sampled based on its frequency in the column and a category with the minority class is not sampled more than a category with the majority one. Finally, a random value is selected according to the PMF and its component in the mask vector associated with the discrete column is set to one, setting all other elements of other mask vectors to zero. Thereafter, the condition vector is constructed by the concatenation of all mask vectors as explained thoroughly above. Figure 3.2 illustrates the CTGAN structure and the training process in detail.

The generator and discriminator (critic) in the CTGAN architecture are two fully connected neural networks with 256 neurons. The generator uses batch normalization and ReLU in the hidden layers while using a mixture of activation functions in the output layer. The numerical part of the continuous columns is generated by the tangent hyperbolic function whereas the cluster vector of the continuous features and the one-hot encoded representations of categorical columns are generated by the Gumbel- Softmax.

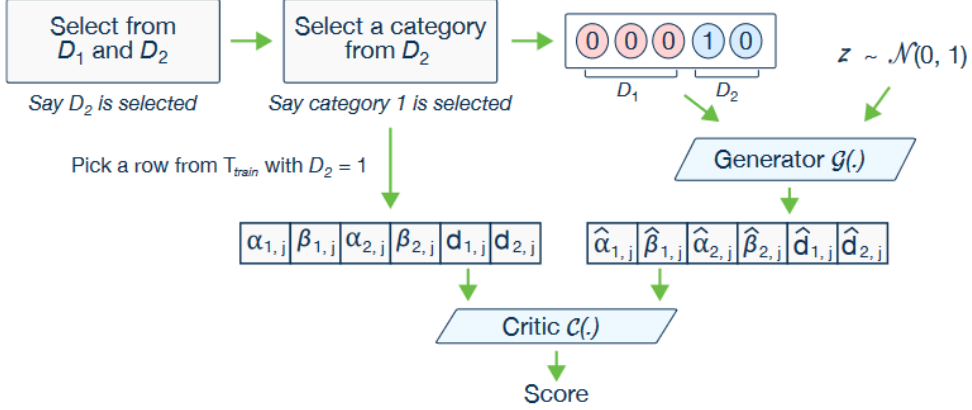


Figure 3.2: An illustration of the CTGAN training process [75].

The structure of the CTGAN generator with the input size of $|cond| + |z|$ is described as [75]:

$$\begin{aligned}
 h_1 &= h_0 \oplus \text{ReLU}(\text{BN}(f_{|input| \rightarrow 256}(z \oplus cond))) \\
 h_2 &= h_1 \oplus \text{ReLU}(\text{BN}(f_{|input|+256 \rightarrow 256}(h_1))) \\
 v_i &= \tanh(f_{|input|+512 \rightarrow 1}(h_2)) & 1 \leq i \leq N_c \\
 u_i &= \text{gumbel}_{0.2}(f_{|input|+512 \rightarrow m_i}(h_2)) & 1 \leq i \leq N_c \\
 d_i &= \text{gumbel}_{0.2}(f_{|input|+512 \rightarrow |D_i|}(h_2)) & 1 \leq i \leq N_d
 \end{aligned}$$

Regarding the discriminator (critic) network, CTGAN authors modified the TGAN version and implemented a packed discriminator as proposed in the PacGAN [41] paper. In simple words, instead of having a discriminator that classifies one sample as real or fake, multiple observations from the same class are simultaneously given to the discriminator, helping to mitigate the mode-collapse issue significantly. Moreover, the leaky ReLU activation function and dropout regularization method are used in the layers of the discriminator. The structure of the discriminator (critic) can be defined as below:

$$\begin{aligned}
 h_0 &= r_1 \oplus \dots \oplus r_{10} \oplus cond_1 \dots \oplus cond_{10} \\
 h_1 &= \text{drop}(\text{leakyReLU}_{0.2}(f_{10|r|+10|cond| \rightarrow 256}(h_0))) \\
 h_2 &= \text{drop}(\text{leakyReLU}_{0.2}(f_{256 \rightarrow 256}(h_1))) \\
 C &= f_{256 \rightarrow 1}(h_2)
 \end{aligned}$$

Where the discriminator considers 10 samples at once ($\text{pac} = 10$). Finally, the CTGAN architecture is adopted from WGAN-GP instead of using the traditional vanilla GAN and same as TGAN, the authors chose ADAM as the optimizer. All these modifications make the CTGAN model much faster and computationally efficient.

3.1.3 TableGAN

In 2018, Park et al.[50] introduced a general-purpose GAN-based model for generating tabular datasets containing different data types (probably the first with such capability). To create synthetic records that are statistically similar to the original ones and to capture inter-variable dependencies between features, their architecture, tableGAN, is developed based on the Deep convolutional neural networks (DCNN). In addition to the generator and discriminator networks, the authors used an auxiliary classifier in their architecture to preserve the consistency and enhance the quality of the synthetic samples. They showed that using such a classifier would prevent the generation of synthetic records that are not semantically correct. For instance, the generation of a patient record with *gender = female* and *disease = prostate cancer* should be avoided because it is logically incorrect and cannot exist in the original table. Moreover, unlike many tabular SDG models, the tableGAN architecture is developed to be protected against *membership inference attacks*. To prevent this attacking scenario, the authors used *hinge loss* in the generator to control the privacy and synthesis quality simultaneously.

Despite other tabular SDG models, the original table is pre-processed straightforwardly in the tableGAN implementation. After transforming categorical values into numerical ones and normalizing both numerical and categorical columns, each record is converted into a 2-dimensional square matrix for usage in the convolutional layers. For instance, a sample with 35 features is transformed into a 6×6 square matrix, including one padded zero.

Regarding the architecture, the generator in tableGAN is a neural network performing multiple deconvolution operations in the subsequent layers, whereas the discriminator is a multi-layer convolutional neural network. Figure 3.3 illustrates these adversarial components in detail. While the generator uses Batch Normalization and ReLU activation function in the intermediate layers, the discriminator implements LeakyReLU in the hidden layers and Sigmoid function in the last layer.

It is worth mentioning that the configuration of each deconvolutional layer in the generator (or convolutional in the discriminator) and the dimension of the input noise should be adjusted regarding the number of features ($16 \times 16 = 256$ in the above illustration).

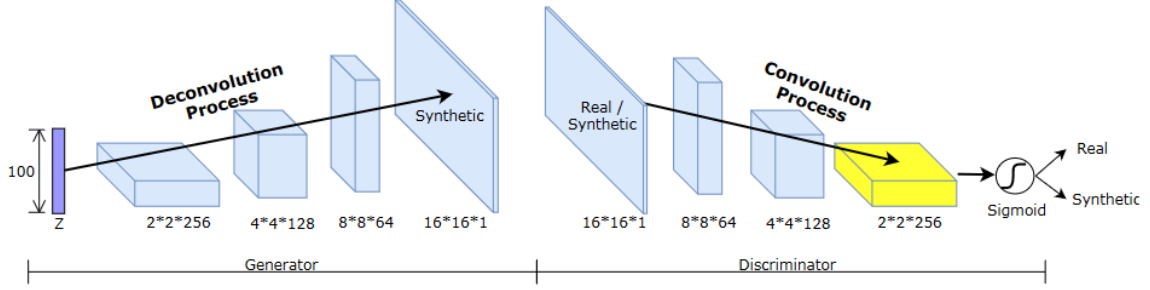


Figure 3.3: An illustration of the tableGAN architecture[50].

The architecture of the auxiliary classifier is identical to the discriminator to enhance the semantic integrity of the synthetic records. The classifier network uses one of the columns from the original data as the ground-truth label for penalizing the model if the generated selection is semantically wrong. Due to the space limitations, the classifier is not included in figure 3.3.

The original paper proposed using three loss functions in the training process: the *original loss*, *information loss*, and *classification loss*. The first one is adopted from the DCGAN architecture representing the conventional adversarial manner between the generator and discriminator. Both adversarial components implement this loss function in their architecture, denoted as L_{org}^G and L_{org}^D respectively. The information loss measures the discrepancy between the mean and standard deviation of the original and generated samples. Hence, the generated records are statistically similar to the original ones. Moreover, to control the privacy degree and quality of the synthetic, the authors proposed using two adjustable hyper parameters (δ_σ and δ_{mean}) to avoid the generation of synthetic records that are statistically too similar to the original records. They also used the concept of the hinge loss to protect the model against membership inference attacks. The information loss L_{info}^G is defined below [50].

$$\begin{aligned}
 L_{mean} &= \|\mathbb{E}[f_x]_{x \sim p_r} - \mathbb{E}[f_{G(z)}]_{z \sim p_z}\|_2 \\
 L_\sigma &= \|\sigma[f_x]_{x \sim p_r} - \sigma[f_{G(z)}]_{z \sim p_z}\|_2 \\
 L_{info}^G &= \max(0, L_{mean} - \delta_{mean}) + \max(0, L_\sigma - \delta_\sigma)
 \end{aligned}$$

Finally, the loss regarding the auxiliary classifier measures the dissimilarity between the label of a synthetic sample and the classifier’s predicted label for the same sample. While the generator in the tableGAN architecture uses a combination of all losses ($L_{org}^G + L_{info}^G + L_{class}^G$) in the training process, the discriminator and classifier are trained by the original loss L_{org}^D and classification loss L_{class}^C , respectively. Furthermore, the tableGAN authors implemented Adam as the optimizer.

3.1.4 CTABGAN

Conditional tableGAN(CTABGAN) is one of the recent deep learning-based architectures developed to overcome the limitations of its prior counterparts in the SDG domain. Specifically, Zhao et al.[79] addressed three challenges that nearly all the previous state-of-the-art architectures faced: modeling a column with mixed data types, a column with long-tail distribution, or a skewed multi-modal continuous feature. To reach these goals and design an effective general-purpose data synthesizer, the core features of the CTABGAN architecture are adopted from the CTGAN and tableGAN models. In other words, the CTABGAN authors proposed combining a conditional generator and the training-by-sampling approach (introduced in the CTGAN model) with the tableGAN architecture. They tried to handle the highly imbalanced features and improve the data generation quality by incorporating the strengths and core features of the CTGAN and tableGAN models.

First, they invented a mixed-type encoder to handle features containing both numerical and missing values or even categorical ones. Often, in real-world datasets, we face variables with a numerical nature having a characteristically categorical value (a value with a specific meaning). For instance, the mortgage column in the Loan dataset contains either 0, indicating no mortgage or any other positive continuous values. The authors argued in the original paper that none of the existing GAN-based SDG models could capture the special meaning of the 0-value, leading to the generation of the negative mortgage values around it. Moreover, there are many examples where a continuous column contains a categorical value indicating the missing values. The authors upgraded the mode-specific normalization introduced in the CTGAN paper and proposed an encoder to circumvent the challenges regarding the mixed-type columns. The encoding process can be detailed based on the mixed-type column distribution illustrated in the figure 3.4a. We can observe that the values are either distributed continuously around two Gaussian distributions (μ_1 and μ_2) or belong to the μ_0 and μ_3 categories. The proposed encoder deals with the continuous part of the mixed-type column, which is similar to the CTGAN’s normalization technique. It uses a variational Gaussian mixture (VGMM) to learn the non-Gaussian distribution part of the column, normalizes each value based on the mode they most likely belong to, and finally concatenates the normalized value with a one-hot encoded vector indicating that specific mode. To deal with the categorical part of the mixed-type columns, they concatenated the values corresponding to each category (μ_0 or μ_3) with a one-hot encoded vector representing their modes. For instance, a value in the μ_0 category is encoded to $\mu_0 \oplus [1, 0, 0, 0]$. Also, the categorical values can include any string or missing values. Furthermore, the categorical columns are encoded to one-hot encoded vectors. It is worth mentioning that the missing values are handled as a different category if they exist in

the categorical features. Finally, the encoder concatenates all the column encodings (continuous/mixed-type and categorical columns) to represent a single row.

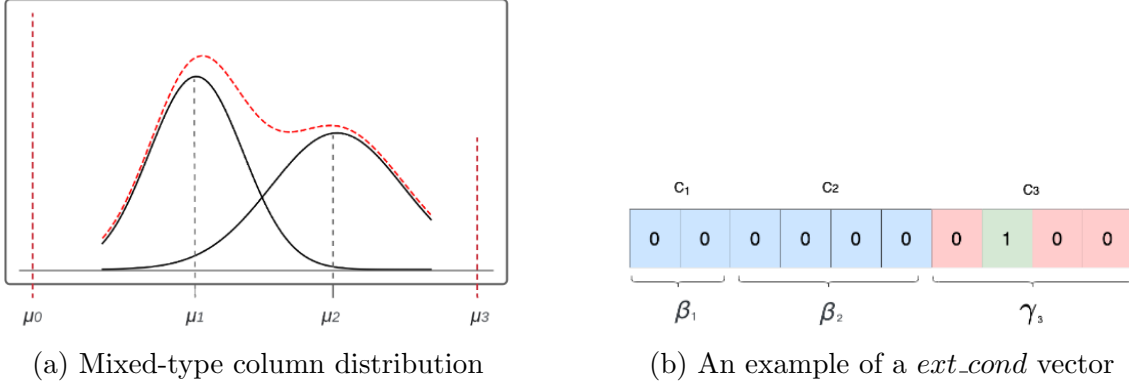


Figure 3.4: The modified encoding and conditional vector [79].

The CTABGAN authors implemented the training-by-sampling and a conditional vector adopted from the CTGAN architecture to tackle generating the imbalanced datasets. However, they modified the CTGAN conditional vector (*cond* vector) to include all one-hot encodings associated with continuous, mixed-type, and categorical columns. Figure 3.4b shows an example of this extended conditional vector (*ext cond*) used in the CTABGAN model. While the CTGAN model uses mode-specific normalization and the training-by-sampling approach to detect multi-modal continuous columns, the CTABGAN authors showed that the inclusion of the continuous variables’ cluster vector in the conditions would make their model more robust in generating skewed multi-modal features.

Using a VGMM to encode continuous values does not always yield a good result, especially for the long-tailed distributions. The Gaussian mixture model does not correctly represent the values towards the tail of the distributions. To circumvent this issue, the model utilizes a logarithmic transformation to reshape the original distribution into a mixture of Gaussian ones and then uses the VGMM.

Although the classifier in the tableGAN model has the same architecture as the discriminator in the CTABGAN, the CTABGAN model utilizes a 7-layer MLP network to improve the semantic integrity of the generated records. Another distinction between the two models’ classifiers is that the CTABGAN classifier can conduct both binary and multi-class classification, while in the tableGAN, the classifier only performs binary classification.

Finally, the CTABGAN’s training procedure is the same as tableGAN’s, and it is comprehensively detailed in [39].

3.1.5 WGAN

Lastly, we implement a vanilla GAN as a baseline model to examine the proposed SDG architectures in a comparative context. The baseline model¹ is a Wasserstein GAN variation with gradient penalty (WGAN-GP) generating both categorical and continuous features as detailed in section 2.3.3. To normalize numerical columns, the model uses a Standard Scaler to subtract the mean from the values in the queue and scale them to unit variance. On the other hand, the categorical variables are encoded to one-hot representations.

The generator and critic’s structures are fully connected neural networks with 256 neurons using batch normalization, ReLU (only in generator), and LeakyReLU (only in critic) in their internal layers. To generate both data types, the generator uses a tangent hyperbolic activation function for the continuous columns and a Gumbel Softmax function for the categorical ones. The adversarial components’ structures are detailed below.

Generator structure:

$$\begin{aligned}
 h_0 &= \text{ReLU}(\text{BN}(f_{|z|\rightarrow 256}(z))) \\
 \text{for } k &= 1, 2, \dots, n_{gen} \\
 h_k &= \text{ReLU}(\text{BN}(f_{256\rightarrow 256}^k(h_{k-1}))) \\
 c_i &= \tanh(f_{256\rightarrow 1}(h_{n_{gen}})) & 1 \leq i \leq N_c \\
 d_i &= \text{gumbel}_{0.2}(f_{256\rightarrow |D_i|}(h_{n_{gen}})) & 1 \leq i \leq N_d
 \end{aligned}$$

Critic structure:

$$\begin{aligned}
 h_0 &= \text{LeakyReLU}_{0.3}(\text{BN}(f_{|input|\rightarrow 256}(input))) \\
 \text{for } k &= 1, 2, \dots, n_{crit} \\
 h_k &= \text{LeakyReLU}_{0.3}(\text{BN}(f_{256\rightarrow 256}^k(h_{k-1}))) \\
 C &= f_{256\rightarrow 1}(h_{n_{crit}})
 \end{aligned}$$

Where $f_{m\rightarrow k}(x)$ denotes a fully connected neural network with an input layer of size m and an output layer of size k . Moreover, n_{gen} and n_{crit} are the number of layers in the corresponding networks.

Finally, the model uses the Adam as optimizer in the training process.

¹https://github.com/TVSjoberg/gan-thesis/tree/master/gan_thesis/models/wgan

3.2 Evaluation Framework

One of the challenges of the SDG is evaluating the diversity and fidelity of the generated samples, independent of whether they are images or records in a tabular dataset. The choice of a proper evaluation framework can often lead to the development of GAN models or even designing new ones. So far, there is a plethora of publications regarding synthetic healthcare data generation, however, the evaluation metrics used in the different models vary significantly and there is no consensus over a single evaluation metric that can cover all aspects. Theis et al. [63] demonstrated that different evaluation metrics could yield significantly different results and consequently, the evaluation metrics should be selected based on the applications of the GAN model. In other words, a good performance with one evaluation metric cannot guarantee a good performance by choosing other evaluation metrics. Furthermore, a survey regarding the generation of tabular health data shows that the proposed GAN papers from 2017 to 2020 have utilized different evaluation metrics [12]. Generally, the metrics can be divided into three categories: *statistical resemblance (general utility)*, *Machine Learning utilities (Specific utility)* and *privacy metrics*. The general utility metrics measure the statistical difference between the original and fake data, while the specific utility metrics compare the similarity of results of certain models trained on both real and synthetic data [61]. Lastly, the privacy metrics quantify the preserved privacy of the real data in synthetic data generation process to assess how much sensitive and private information of the real dataset is compromised and leaked when generating new data samples. In this thesis, we make use of a combination of all the evaluation categories mentioned and provide a comprehensive explanation of each one in the following.

3.2.1 Statistical Resemblance (General Utilities)

This group of evaluation metrics measure the distributional similarity between the original and synthetic data to make sure that the GAN models have preserved the statistical properties of the real dataset. Multiple statistical metrics and distances are utilized to quantitatively measure the resemblance between real and synthetic data [5, 36, 42].

Basic Statistical Check

First, we compare the simple statistics of the original and generated data to see if the mean and standard deviation of all the features have changed considerably or not. One way of doing this is to plot the means or standard deviation of each feature on the same figure while x-axis belongs to the real data and y-axis goes with the synthetic data. Alternatively, the author of [5] proposed a way that aggregates basic

statistics of real and synthetic data based on the formula below. He demonstrated that by applying the Spearman’s ρ correlation on the concatenated lists of the means and standard deviations of each column for the real and fake data could yield a single numeric value that is an indication of the extent to which the basic statistics are preserved. This value can be calculated by:

$$\mathcal{S}_{basic}(R, F) = \rho_{spearman} (\langle mean(real), std(real) \rangle, \langle mean(fake), std(fake) \rangle)$$

In the above formula, $\langle A, B \rangle$ is the concatenation operator and $mean(real)$ refers to the list of means of each column of real dataset.

Column Distributions

In this step marginal distributions of numerical and categorical columns in real and synthetic data are compared to ensure that the model has preserved the column distributions in the generation process or has faced failure modes like the mode-collapse. This evaluation step can be done in two ways. Either by visual comparison of PDF, PMF or CDF plots of each column in the real and synthetic dataset [5] or by conducting statistical tests such as *Kolmogorov-Smirnov* and *Chi-squared* tests on each feature separately. As proposed in [47], two-sample Kolmogorov–Smirnov test is applied on all numerical columns to find the maximum distance between the cumulative distribution functions of two numerical columns in the real and synthetic dataset and it returns the average of one minus the Kolmogorov–Smirnov test D statistic values across all numerical features. On the other hand, for categorical features, the probability mass functions are compared by Chi-squared test [47]. In both cases, if the marginal distributions of the real and synthetic columns are the same, the output of these statistical tests will be one. otherwise, the outcome will be zero if they are completely different.

Column-wise Associations

To examine how well the interactions and dependency structures of the columns are captured in the synthetic data generation process (how well the columns are correlated to each other), we must compare the column correlations of real and synthetic data [5, 80]. In this way, we can tell which feature relationships are preserved in the SDG process and which are not. The definition of correlation is often intertwined with Pearson’s R coefficient; however, this metric is not applicable for categorical columns. One of the SDG challenges we face both in the generation process and this evaluation step is working with a mixed tabular data containing different data types like categorical, binary, continuous etc which prevent us from using the Pearson’s

coefficient for the whole dataset. One way to circumvent this issue is using *One-hot encoding* to convert the categorical columns into numerical ones. Consequently, we can use Pearson’s coefficient to calculate correlation. However, one-hot encoding grows the size of the dataset significantly (a dataset with 20 categorical columns is turned into a new dataset with 130 columns) and makes interpretation of the correlation matrix demanding and, in many cases, impractical. Alternatively, [80] proposed to use a measure of association for categorical features instead of using a single Pearson’s correlation coefficient for all types of features. He argued that pairwise columns should be split into distinct classes based on their data types and each one should apply a different association metric to capture the correlation. The proposed metrics are detailed below.

Numerical-Numerical

When both columns are numerical, the *Pearson’s r* correlation coefficient [71] is a trivial choice of the association metric.

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

It ranges between -1 and 1 where negative values indicate negative correlation between columns.

Categorical-Categorical

In the case of two categorical features, [80] proposed two ways of measuring association between columns. *Theil’s U (Uncertainty coefficient)* [72] and *Cramer’s V* [70] coefficients. The author argued that due to the asymmetric nature of the Theil’s U ($U(X, Y) \neq U(Y, X)$) it is the preferred choice compared to the Cramer’s V metric (Cramer’s V is symmetrical metric which results to the loss of valuable information). Theil’s U is based on conditional entropy, and it ranges between 0 and 1 where 0 means y provides no information about x [80]. The Uncertainty coefficient can be formulated as:

$$U(X|Y) = \frac{S(X) - S(X|Y)}{S(X)}$$

where $S(X)$ and $S(X|Y)$ denote the entropy of X and the conditional entropy of X given Y respectively.

Numerical-Categorical

A proper choice of association metric in the case of mixed type columns is *Correlation Ratio* η [69]. Mathematically, it is computed by the formula below and it ranges between 0 and 1, where higher values are indication of larger association. In a simple word, given a numerical value, this metric measures the information gain of the category it can be belong to.

$$\eta = \sqrt{\frac{\sum_c n_c (\bar{y}_c - \bar{y})^2}{\sum_{c,i} (y_{ci} - \bar{y})^2}}$$

In the above formula, n_c stands for the total observations in the category c and \bar{y}_c and \bar{y} are defined as:

$$\bar{y}_c = \frac{\sum_i y_{ci}}{n_c} \quad , \quad \bar{y} = \frac{\sum_i n_c \bar{y}_c}{\sum_c n_c}$$

Using the selected metrics, the author of [80] designed the *dython*¹ library for the calculation of the association matrices between the tabular datasets. We used this library in our evaluation framework to compute the association matrices of the real and fake datasets and calculate the overall distance between them on an element-wise basis.

¹<http://shakedzy.xyz/dython/>

3.2.2 Specific Utilities (Machine Learning Utilities)

Machine Learning-based Detection

To assess how indistinguishable the generated records are from the real ones, we train classifiers on all the records from the real and synthetic datasets, labelled as real or fake, and evaluate the SDG based on the models' detection capability [42, 47]. Thus, a poor classification result indicates that the SDG process was successful enough and the real and synthetic data is hard to be separated. We use Logistic Regression and Support Vector Machines (SVM) classifiers as the machine learning detection models and choose the area under the ROC curve (AUROC score) as the performance metric of the classification task. If the synthetic data is inseparable from the original one, the AUROC score would be 0.5, indicating that the classifier is guessing randomly and unable to distinguish the real and fake classes. The less the AUROC score of the classification, the better the synthetic samples have been generated. However, since most of the evaluation metrics in our setting are in the range of 0 to 1, we normalize the classification result to one minus the average AUROC score with 1 indicating the original and synthetic datasets are completely inseparable.

Machine Learning-based Cross-Testing

Of all the metrics proposed for the evaluation of tabular GAN models, machine learning-based cross-testing seems to be the most recurring evaluation method in the related papers. The idea is to compare the inferential ability of a machine learning model trained on both the original and synthetic data. More specifically, we split the original data into the training and test sets (X_{train} and X_{test}). After training the generative models on the training set, we generate synthetic data (X_{fake}) of the same size as X_{train} . Now we use X_{train} and X_{fake} separately to train a set of machine learning prediction models and compare their predictive capability on the X_{test} . If the prediction results of the models trained on X_{train} and X_{fake} are significantly close, it indicates that these datasets are equivalent, and we can use the synthetic data instead of the real data in our analysis. If we are trying to make a prediction based upon a binary or categorical column, we are doing classification and the machine learning models must be classifiers; otherwise, they are regressors. Since all the healthcare datasets studied in this thesis include a binary or categorical target column, a set of classifiers is chosen to compare their predictability on the real and synthetic data: Decision Tree, Random Forrest, Logistic Regression, and Multi-layer Perceptron classifier. To evaluate the effectiveness of these classifiers, the harmonic mean between the precision and recall (F1-score) is calculated as the performance measure. It is worth mentioning that the goal here is not to train the best ML predictors or to tune the models' hyper-parameters to get the best

prediction score. Instead, our intention is to investigate if the results for the real and synthetic data are nearly the same when the ML predictors are equally tuned. Consequently, all hyper-parameters use their default value, and all ML prediction models are fixed for all case studies [36, 42]. Figure 3.5 illustrates the steps in this specific utility measure.

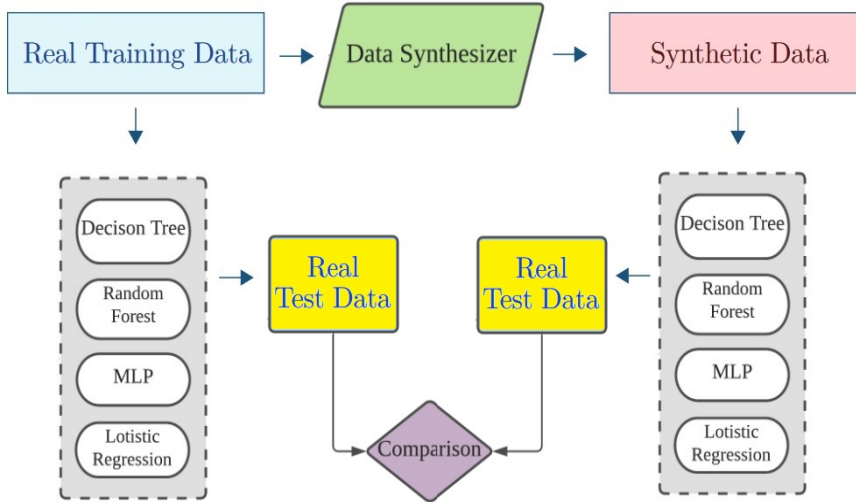


Figure 3.5: The workflow of cross-testing evaluation

3.2.3 Preserved Privacy

In addition to the general and specific utility metrics elaborated so far, privacy metrics are the last and most crucial evaluation category the data science practitioners need to consider for protecting the individual’s privacy, especially in the healthcare domain. After the generation of synthetic data, if an individual’s private data in the real dataset is to be re-identified in any way, it indicates that sensitive information associated with the individual is included in the synthetic dataset and his privacy has been compromised. Thus, we need a mechanism to measure the patients’ privacy. Broadly speaking, the privacy evaluation of the GAN models is divided into two categories: *Theoretical evaluation* and *Empirical evaluation* [77]. The first category defines privacy mathematically and guarantees that this theoretical concept is satisfied in the SDG algorithm. Differential Privacy (DP) [14], as an example, is one of the most recurring privacy evaluation guarantees in many research papers that belongs to this category. In simple words, this mathematical concept ensures that if the outcome of a query of two datasets is not significantly affected by the inclusion of an individual data, the chance of privacy breach is small, and it is unlikely that the individual’s identity is exposed if his sensitive information is included in the dataset [15, 68]. Although it is theoretically proven that DP can act as a protection layer

against privacy attacks such as *Linkage attacks* and multiple papers have already used its properties in their GAN settings [34, 73, 78], it causes a significant decrease in the data utility.

On the other hand, the empirical evaluation metrics are the most common privacy metrics investigated in the related literature and are conducted in two ways: either by the examination of the GAN’s generator or the synthetic generated data. To measure how much sensitive information is leaked by the GAN generator, [11, 50, 65] tested their proposed models against the *membership inference attacks* scenario (formerly known as presence disclosure attack) to check if the generator is vulnerable or not. In this scenario, the attackers often have black-box access to the model’s architecture and their goal is to determine whether a sample is included in the training data or not, after the observation of the outputs of a target machine learning model [26, 60]. In contrast to the measurement of the preserved privacy of the model’s generator in attacking scenarios, the quantification of the privacy level in the synthetic generated data is more intuitive and easier to grasp for data scientists. Consequently, to measure preserved privacy, many related publications (including this thesis) use *Distance-based* metrics to quantify the degree of privacy in the synthetic data [50, 74, 79]. Here, to evaluate how similar the real and synthetic data records are, we calculate the *Euclidean* distance between a record of the synthetic data and its closest neighbor in the real data. This way of measuring privacy is referred to as the *Distance to Closest Record(DCR)* [50] and the larger the DCR is, the chance of having a privacy breach is smaller. It is preferred that the distribution of the calculated Euclidean distances has a large mean and a small standard deviation. Note that a large standard deviation indicates the probable existence of many synthetic and real pairs which are too close (too similar), even if the mean of the distances is large enough [5, 50].

The code of all the selected metrics and statistical distances in our evaluation framework have been implemented in multiple sources such as *TableEvaluator*¹, *SDMetrics*², and *dython*³ libraries and we use a combination of them in this thesis.

¹<https://github.com/Baukebrenninkmeijer/table-evaluator>

²<https://github.com/sdv-dev/SDMetrics>

³<http://shakedzy.xyz/dython/>

Chapter 4

Experimental Setup

To evaluate the selected SDG models in the healthcare domain, we perform several experiments with different medical datasets. This chapter provides comprehensive descriptions of each healthcare dataset used in the thesis, followed by the implementation details and parameter choices for each investigated model.

4.1 Data

The initial objective was to find an optimal solution to synthesize medical records based on a confidential *Tacrolimus Exposure* dataset. The data was obtained from four clinical studies performed at Oslo University Hospital - Rikshospitalet, Norway in the period 2011-2018 [24, 25, 45, 56]. This dataset was mostly numerical and contained data records regarding the tacrolimus dosages exposed in the patients' bodies in different time stamps. After aggregating each patient's time-series records into a vector to transform the original data into a tabular format, it was observed that the size of the resulting dataset became significantly small for any SDG models (we came up with data records of fewer than 100). To synthesize records with high quality and diversity, we often need a comprehensive, large-sized training dataset so the synthetic generative models can learn the underlying distributions of the features and capture inter-variable relationships between variables to generate useful, appropriate synthetic records. Otherwise, the model would overfit and simply memorize the training data, resulting in the generation of records that look exactly like the ones in the training set. Consequently, we decided to exclude this case study from our experiments and test the chosen generative models on other publicly available medical datasets containing enough data records and more complex inter-variable relationships. We selected several healthcare datasets from multiple sources to com-

pare the efficacy of synthetic generative models: *Diabetes*¹ and *Thyroid Disease*² datasets from UCI Machine Learning Repository, *Epileptic Seizure Recognition*³ dataset from Kaggle, and MIMIC III⁴ clinical database from *PhysioNet* platform.

Due to the inclusion of variables with different data types and feature distributions in these datasets, it is likely to tell which generative model works better for data with specific modalities. All datasets have a binary or categorical target variable for which we can conduct classification tasks. Moreover, no feature engineering or scaling has been done on the original version of these datasets because each model has its own way of input data transformation which was elaborated in detail in each generative model’s subsection. We only conducted some simple pre-processing tasks to assure that the input data variables do not contain any mixed data types or null values. It should be mentioned that some models like CTABGAN are capable of dealing with mixed-type features or handling null values, while others are not. Since the aim of the thesis is to conduct a fair comparison of the SDG models in the healthcare domain, not synthesizing data with specific characteristics, we should prepare the data in a proper, acceptable manner for all the generative models we are testing. The description of each dataset is presented in the following and the overall statistics associated with them are summarized in table 4.1.

4.1.1 Epileptic Seizure Recognition

This dataset consists of the *Electroencephalogram* (EEG) recordings of 500 patients. An EEG is a test to diagnose any abnormality in the electrical activity of the human brain. It is often used to diagnose *Epileptic Seizures*, one of the most common brain disorders. Each of these EEG time series is a recording of a patient’s brain for 23.5 seconds. Then each recording is sampled into 4097 points and subsequently, the points are divided into 23 chunks. Thus, each chunk represents 1 second of the time series and it contains 178 data points. As a result, the EEG time series are converted into a tabular dataset with $23 \times 500 = 11,500$ rows and 178 columns. The target variable in the last column indicates whether a patient faces epileptic seizures or not. The values in this column range between 1 to 5, while subjects with class 1 had seizures and subjects with other classes did not. As an example, class 4 and class 5 indicate that while recording the EEG, the patients had their eyes closed and opened respectively. Thus, we can perform both binary and multi-class classification on this dataset. Since the Epileptic Seizure Recognition dataset provided by Kaggle is a pre-processed version of the original one [2], no extra pre-processing is needed.

¹<https://archive.ics.uci.edu/ml/machine-learning-databases/00296/>

²<https://archive.ics.uci.edu/ml/datasets/Thyroid+Disease>

³<https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition>

⁴<https://physionet.org/content/mimiciii/1.4/>

4.1.2 Diabetes

The Diabetes dataset was the outcome of a study [62] conducted on the *Health Facts* database, a comprehensive clinical database in the United States. The Health Facts database consists of ten years of clinical data at 130 US hospitals across the United States. The aim of the study was to investigate whether the use of *Hemoglobin A1c* (HbA1c) as a measurement could lead to a reduction in the readmission rate of the patients diagnosed with diabetes [62]. The study resulted in the extraction of the Diabetes dataset from the Health Facts database containing the diabetic inpatient encounters. To be more specific, any inpatient with any kind of diabetes who stayed in the hospital for a minimum of 1 day and a maximum of 14 days, received medications and underwent laboratory tests is included in the dataset. The extracted dataset is composed of 101,766 rows (inpatient encounters) and 50 columns (that are potentially related to the diabetic condition of a patient) including age, gender, race, diagnoses, diabetic medications, lab measurements, admission type, etc. The *readmitted* column as the target variable indicates whether the patient’s readmission occurs within or after 30 days of discharge from the hospital, or there won’t be any readmission at all. Thus, we can perform both multi-class classification and binary classification (if we only consider readmission or no readmission) on this dataset.

Pre-processing

As with any real-world dataset, there are incomplete and noisy records in the preliminary version of the diabetes dataset. First, the *weight*, *payer code*, and *medical specialty* columns are removed due to the inclusion of a high percentage of missing values (“?” character in this dataset). We also decided to simply drop all the rows containing missing values instead of replacing them (due to many available data records). The *Encounter ID* and *Patient Number* features are unique identifiers that can be created trivially without any inter-variable relationships between other columns, thus they are dropped as well. There are 24 features associated with specific drugs, indicating whether the medication is prescribed or there is a change in their dosage. By counting unique values in each of these columns, we observe that there is only one value per column for 14 of these features. Since there is not any valid reason to generate columns with only one value, these features are eliminated from the dataset. Consequently, we have a pre-processed subset of the diabetes dataset with 89053 rows and 29 columns ready to be used for the SDG task.

4.1.3 Thyroid Disease

The Thyroid Disease [13] dataset contains the latest version of the thyroid diagnosis records provided by the Garvan Institute of Medical Research in Sydney, Australia. The original dataset consists of 9172 inpatient records and 20 columns gathered from 1984 to 1987.

Pre-processing

After replacing the “?” characters with the Null value, we observe that a high percentage of the *TBG* column is composed of the missing values. Thus, we drop both the *TBG* column and its associated one *TBG measured*. The target variable in this dataset is a string of letters indicating the diagnosed thyroid conditions. These conditions are divided into several groups where each one is associated with a different class of thyroid diagnosis. By extracting the first letter of each string we can tell if a patient is diagnosed with thyroid disease (alphabetical letters starting from A to T) or not having the thyroid disease (“-” letter). We convert the target variable into a binary feature where 1 indicates a patient with thyroid disease and 0 otherwise. Lastly, we fill the missing values in the remaining columns using the simple imputation method from the *sklearn* [51] library.

4.1.4 MIMIC III

Medical Information Mart for Intensive Care (MIMIC-III) [33] is a large, publicly available database containing longitudinal medical records of over 46,000 patients admitted to the *Intensive Care Unit* (ICU) of Beth Israel Deaconess Medical Centre between 2001 and 2012. Different types of de-identified health-related information are included in the MIMIC-III clinical database such as demographics, clinical measurement, laboratory results, medications, diagnoses data, performed procedures, length of stay, mortality, and etc. Although the database is widely available, the researchers are required to complete the “MIT Data or Specimens” training course and formally go through an approval process [19]. Moreover, the patients’ sensitive information in the database is de-identified with classical anonymization methods to ensure patients’ privacy in accordance with the HIPAA regulations. The clinical database investigated in this thesis and the other related studies is a companion to the MIMIC-III Waveform Database consisting of thousands of vital physiological signals (time-series) acquired from each patient’s bedside monitor. The figure 4.1 illustrates how the MIMIC-III database is acquired from different sources with different data types. Since we are focused on the generation of tabular healthcare data (not the generation of time-series data), we exclude the waveform database from further analysis.

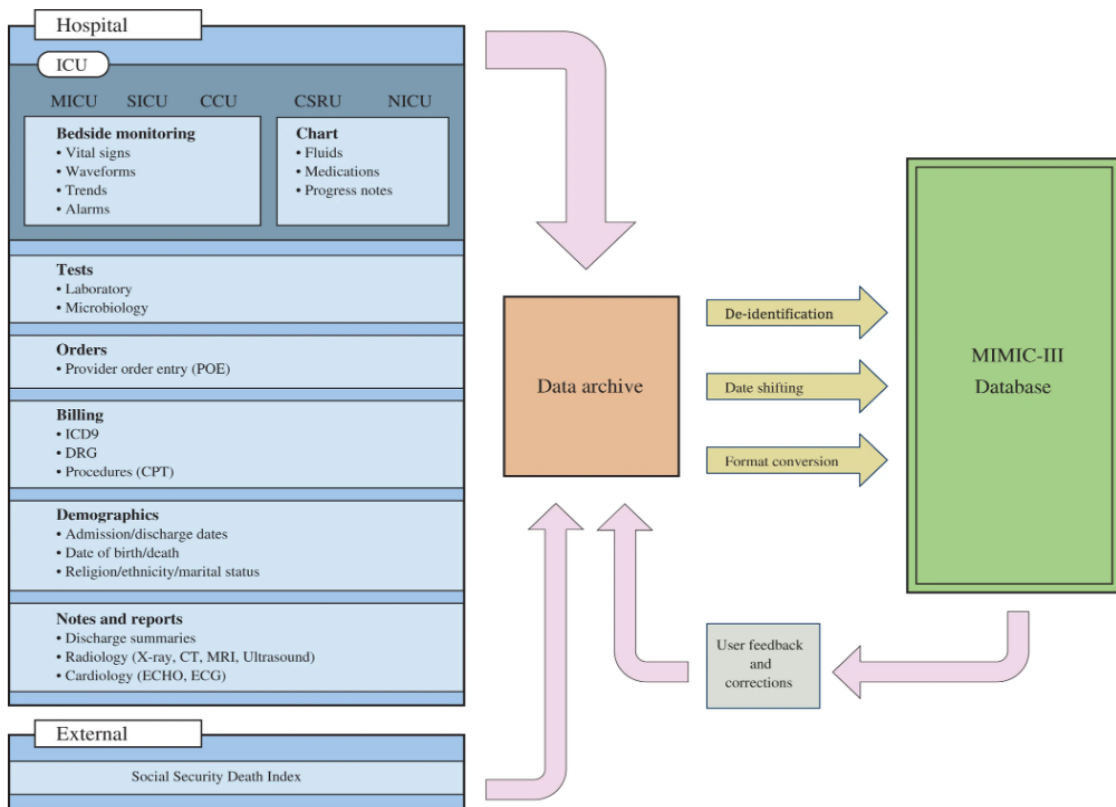


Figure 4.1: An overview of the MIMIC-III database collection [33]

MIMIC-III clinical database is composed of 26 tables which can be joined through four different identifiers: *Subject ID* – a unique integer for each patient, *Hospital admission ID* – a unique integer for each patient’s hospital admission, *ICU stay ID* – an integer identifying a patient’s stay in the ICU, and *Case ID* indicating a set of signals associated with each patient. During the collection of the MIMIC-III clinical database, patients may be readmitted several times and each hospital admission can be associated with multiple ICU stays. Thus, it is crucial to know how to identify patients or track their stays. An overview of the comprising tables and their descriptions is described in the original paper ¹ and the schema of this relational database is provided in the Appendix.

Many of the papers investigating the SDG in the healthcare domain use the MIMIC-III clinical database to synthesize patients’ diagnostic codes (ICD-9 codes). However, we intended to extract a multi-modal table to test the SDG abilities of our selected models on different data types, not just ICD-9 codes.

¹<https://www.nature.com/articles/sdata201635/tables/5>

Pre-processing

We start by combining the *PATIENTS* table, which contains all the unique individuals admitted to the hospital, with the *ADMISSIONS*, *INPUTEVENTS*, *OUTPUTEVENTS*, *SERVICES*, *LAB EVENTS*, and *ICU STAYS* tables. The resulting table (after conducting multiple modifications and pre-processing practices including column dropping, dropping the missing values, replacing values with minority class, etc) contains 40,895 rows and 14 columns. Each record is associated with a unique patient with their demographic information and aggregated medical measurements and test results during their multiple stays in the ICU. The target variable in the extracted table is *HOSPITAL EXPIRE FLAG* indicating whether the patient is healed when discharged from the hospital or passed away while hospitalized. Although the dates are shifted into the future for privacy-preserving reasons in the original database, the duration remain intact. Meaning that by subtracting two consecutive dates, we can achieve the length of the period between them. This is how we come up with the length of the patients' stays in the ICU and hospital. It is worth mentioning that when calculating each patient's age by subtracting the *date of admission* from the *date of birth*, we observe that some patients come up with the age of 300 or more. This happens because the date of birth of the patients with more than 90 years is shifted to hide their true age. To circumvent this, we approximate these patients' age by randomly choosing values between 90 and 100 and assigning probability weights in a decreasing manner. In more detail, under our assumption, it is more probable to have a 90-year-old patient than a patient with the age of 100 years.

Dataset	#Train/Test	Target Column	#C	#BC	#MC
Epileptic Seizure	9,000/2,500	y	178	0	1
Diabetes	70,000/19,000	readmitted	11	3	15
Thyroid	7,100/2,000	binaryClass	6	21	1
MIMIC III	31,900/9,000	Hospital_Expire_Flag	6	2	6

Table 4.1: Summary of the medical datasets used in the thesis. #C, #BC and #MC denote the number of continuous, binary and multi-categorical columns respectively.

4.2 Implementation Details

In this thesis, we conduct our tabular SDG experiments using TGAN¹, CTGAN², CTABGAN³, and WGAN-GP models. TableGAN is not included in our experiments because it seriously suffers when the target variable is multi-categorical (all our medical use-cases). There seems to be a technical issue in the original implementation⁴ that increases the auxiliary classifier’s loss uncontrollably if the target column is not binary. Instead, the CTABGAN model is selected to handle both binary and multi-categorical target features.

All the experiments are conducted using Python 3.8 as the primary programming language. The proposed tabular SDG models are implemented using Tensorflow, except the CTABGAN model built with Pytorch. The implementations regarding each tabular SDG model and the selected evaluation framework can be found in the thesis GitHub repository⁵. Furthermore, The experiments were conducted on the University of Stavanger’s GPU cluster (Gorina6) on an Nvidia Tesla V100 machine equipped with 32 GB of memory. However, the evaluations and comparisons are conducted on a Desktop PC with specifications of AMD Ryzen 5 5600G with 8 GB of memory.

To conduct a fair comparison, all tabular SDG models are trained for 300 epochs for Diabetes and MIMIC III datasets. In contrast, for the Epileptic Seizure and Thyroid tables, the algorithms are trained for 1,000 epochs. Epileptic and Thyroid datasets have significantly fewer records than Diabetes and MIMIC III tables. Consequently, to have an unfavorable comparison, they need more epochs to reach convergence. Each SDG model uses a batch size of 256, an input noise vector of 128, and the learning parameters based on the table 4.2. The hyper-parameter choices and architecture of each model are described as follows.

The generator in the TGAN model consists of an LSTM network with 400 units and a hidden layer of the length of 100. The critic is a 2-layer feed-forward network with 200 units per layer. Moreover, the Gaussian noise added to the categorical variables has an upper bound of 0.2.

The CTGAN and WGAN models have fully connected neural networks for both the generator and the critic networks. Each network has two hidden layers with 256 neurons. Based on the original WGAN paper, the critic is updated five times for each generator update.

¹<https://github.com/sdv-dev/TGAN>

²<https://github.com/sdv-dev/CTGAN>

³<https://github.com/Team-TUD/CTAB-GAN>

⁴<https://github.com/mahmoodm2/tableGAN>

⁵<https://github.com/Ali-HZN/thesis>

In the CTABGAN model, the generator and discriminator are implemented by a four-layer and two-layer CNN, respectively. To decide the dimensionality of the hidden layers for the discriminator and generator networks, the number of channels argument is selected to its default value of 64. Moreover, the auxiliary classifier network is a four-layer fully connected neural network with 256 neurons in each layer.

Finally, a synthetic dataset with the same number of records as the original one is generated for each proposed method.

SDG Model	Learning Rate	β_1	β_2	Weight Decay
TGAN	10^{-3}	0.5	0.9	10^{-5}
CTGAN	2×10^{-4}	0.5	0.9	10^{-6}
CTABGAN	2×10^{-4}	0.5	0.9	10^{-5}
WGAN-GP	10^{-4}	0.5	0.9	0

Table 4.2: Adam optimizer hyper-parameters in each SDG model.

Chapter 5

Evaluation & Results

In this chapter, we evaluate the results of the proposed SDG models based on the evaluation framework presented in Section 3.2. Specifically, we compare the statistical resemblance between the original and synthetic healthcare datasets in Section 5.1, evaluate the machine learning-based similarity scores in Section 5.2, and discuss the level of compromised privacy in Section 5.3. The purpose is to shed light on the strengths and weaknesses of each tabular SDG model in different healthcare use cases. Although we provide a thorough discussion for each model in each dataset, only a subset of visuals is included due to the space limitations. It is worth mentioning that all the evaluated results and visuals for each dataset can be found in the Appendix.

5.1 Statistical Resemblance

5.1.1 Basic Statistical Check

In the first step of evaluating each model, we investigate if the simple statistics of the real healthcare datasets are maintained through the generation process. To do this, the mean and standard deviation of the numerical features of each dataset are visually inspected in a separate, log-scaled figure, while the X-axis represents the original data, and the y-axis represents the synthetic ones. If an SDG model cannot preserve these fundamental properties, it will also struggle with capturing more complex ones.

In the MIMIC III dataset (figure 5.1), most of the blue dots in the first row lie on the diagonal line or in its periphery. This indicates that all proposed SDG models preserve the means of the numerical features with relative ease. Similarly,

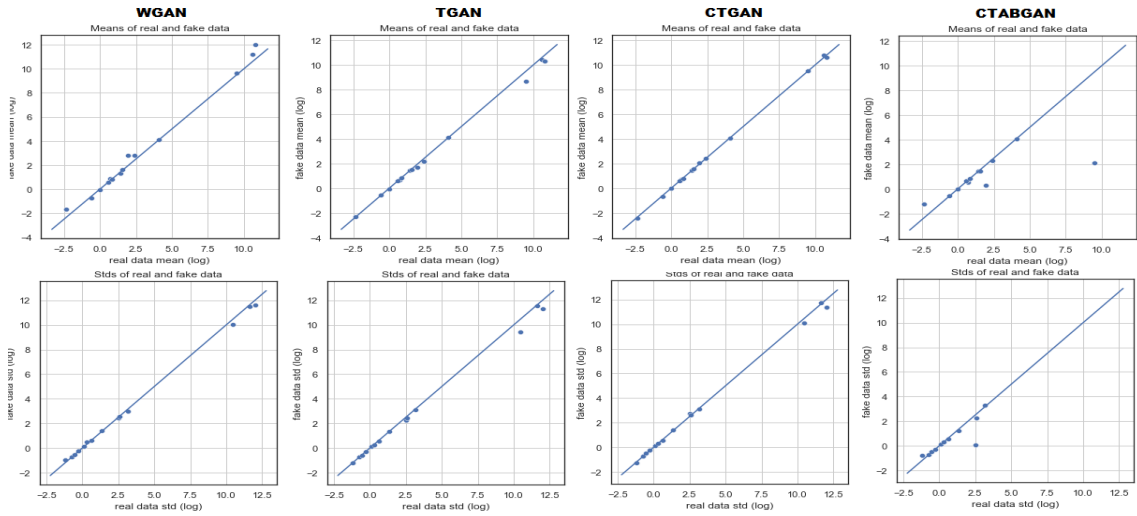


Figure 5.1: The log-transformed means and standard deviations of the numerical columns in the original and synthetic MIMIC III datasets.

the standard deviations of the MIMIC III numerical columns are captured in the generation process, as observed in the second row of figure 5.1. Moreover, we can compute a basic statistical coefficient between the original and synthetic datasets based on the procedure described in Section 3.2.1. This numeric score indicates the extent to which the mean and standard deviation is preserved through the generation process and can be used instead of previous visual inspections.

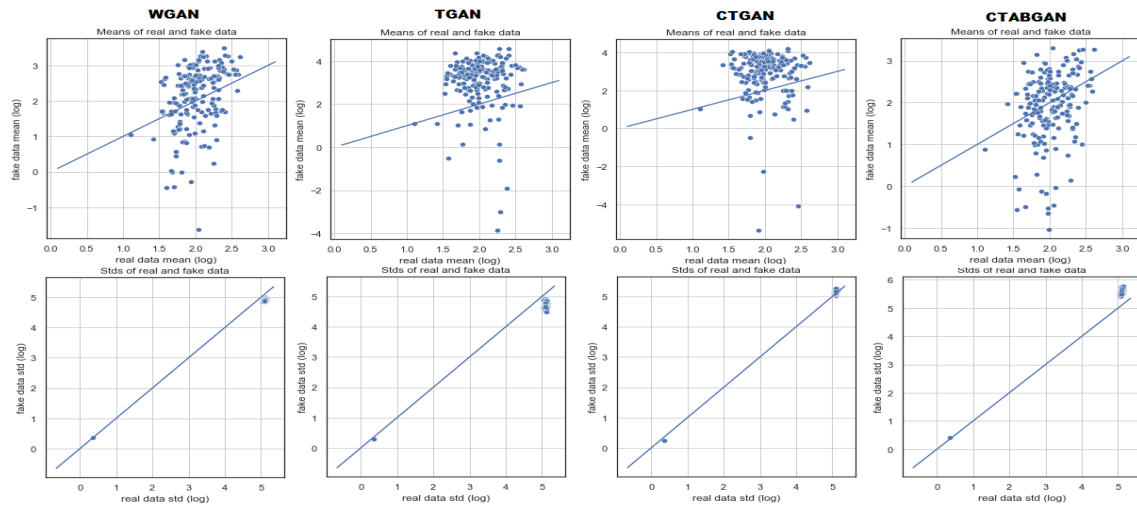


Figure 5.2: The log-transformed means and standard deviations of the numerical columns in the original and synthetic Epileptic datasets.

As for the MIMIC III dataset, the table 5.1 shows that all proposed tabular synthesizers have significantly high coefficients in the Diabetes and Thyroid datasets and capture the mean and standard deviation of their numerical columns with ease.

However, this basic coefficient faces a decrease in the Epileptic dataset compared to others. The behavior can be visually verified in figure 5.2 as well. Although most of the models preserve the standard deviations, we observe much more differences between the means of the original and synthetic Epileptic datasets. The reason why it is much harder to generate a synthetic Epileptic dataset is due to its nature. The dataset comprises 178 discrete numerical features, ranging between -1850 and 1750. None of the proposed SDG models explicitly differentiate between the generation of discrete (integer) and continuous (floating) numerical columns, and, as comprehensively discussed later, the performance of the SDG synthesizer drops significantly when it is trying to generate an integer distribution with a wide range.

	Diabetes		MIMIC III		Thyroid		Epileptic	
	Basic	corr	Basic	corr	Basic	corr	Basic	corr
WGAN	0.960	1.00	0.975	1.09	0.962	1.1	0.875	6.9
TGAN	0.948	0.87	0.967	0.65	0.981	0.55	0.853	17.8
CTGAN	0.987	1.29	0.978	0.95	0.970	1.65	0.835	26.5
CTABGAN	0.976	1.65	0.951	2.13	0.971	2.09	0.868	19.8

Table 5.1: Basic statistical coefficients and absolute correlation distances between the original and synthetic healthcare datasets.

Briefly, all proposed tabular SDG models demonstrate outstanding performance in capturing the means and standard deviation of the numerical columns in the Diabetes, MIMIC III, and Thyroid datasets. However, a gradual decrease in the performance is observed in the Epileptic dataset due to the smaller dataset size and the discrete numerical nature of the columns.

5.1.2 Column-wise Associations

To demonstrate how well the inter-variable dependencies are preserved in the SDG process, we calculate the absolute distance between the original and synthetic association matrices for each use case. According to table 5.1 and figure 5.3, it can be noted that the TGAN architecture outperforms other models in terms of capturing feature interactions with the column-wise absolute distance of 0.87. WGAN performs slightly worse while CTGAN and CTABGAN come in the third and fourth places, respectively.

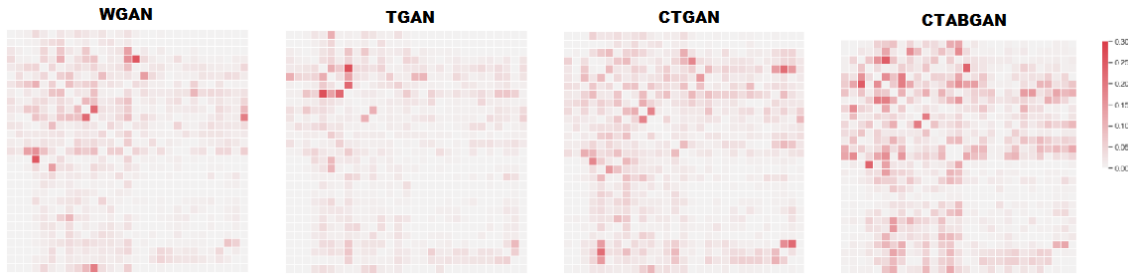


Figure 5.3: Difference between the real and fake Diabetes association matrices.

A similar pattern is observed in the MIMIC III and Thyroid datasets, with the TGAN model performing the best, WGAN and CTGAN models in the following places and the CTABGAN with the worst performance. The reason why TGAN outperforms other models in terms of maintaining column-wise correlation can be attributed to the use of an LSTM network in the generator and synthesizing variables in sequential order. On the other hand, it seems that utilizing the DCGAN architecture and convolutional operations in the CTABGAN’s generator results in poor preservation of inter-variable correlations in these datasets.

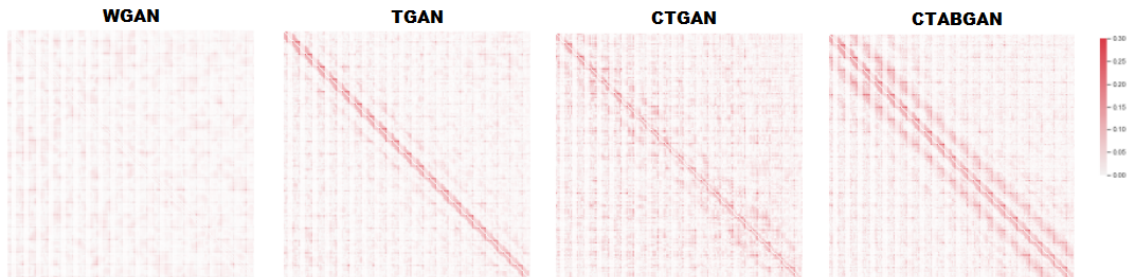


Figure 5.4: Difference between the real and fake Epileptic association matrices.

The results for the Epileptic dataset demonstrate a different pattern compared to the results for the other datasets. In the Diabetes, MIMIC III, and Thyroid datasets, it is noted that the absolute correlation distances for WGAN and CTGAN are slightly close to each other (due to the similarity of their generator networks). However, in the Epileptic dataset, the WGAN model has the best performance in capturing correlation with the distance of 6.9, and CTGAN performs the worst with an absolute distance of 26.5. This is mainly related to the Mode-specific normalization incorporated in the TGAN, CTGAN, and CTABGAN’s pre-processing step. Although this normalization step works well for complex datasets, the associated increased dimensionality of the original data leaves no room for models to capture correlations effectively. Figure 5.4 illustrates the absolute difference between the original

and synthetic Epileptic correlation matrices. It can be observed that besides the best and worst models, TGAN and CTABGAN come in the second and third place respectively. The real, fake, and absolute differences in association matrices of each dataset are included in the Appendix.

In a nutshell, the TGAN model outperforms others in capturing the correlations in the Diabetes, MIMIC III, and Thyroid datasets. This behavior demonstrates that generating each record sequentially in the TGAN generator is the best way to capture inter-variable relationships in these datasets. However, in the Epileptic dataset, the WGAN outperforms TGAN due to the increased dimensionality and the nature of the original data.

5.1.3 Column Distributions

To compare the marginal distributions of the columns in the original and synthetic datasets, we use the Kolmogorov–Smirnov test and Chi-squared test between pairwise numerical and categorical columns, respectively, and report the average score for each column type. For numerical columns, the test result is one minus Kolmogorov–Smirnov D statistic, and for categorical ones, the outcome is the resulting p-value. In both cases, if the probability distributions or probability mass functions of two columns are the same, the test result is one. Otherwise, the resulting score is zero if the distributions are completely different. The results of these statistical tests are reported in table 5.2.

In the Diabetes dataset, CTABGAN outperforms other models in preserving the marginal distributions of the numerical columns with the Kolmogorov–Smirnov test score of 0.90, while the TGAN and CTGAN models are slightly worse, with a score of 0.87. Although CTGAN, TGAN, and CTABGAN benefit from the Mode-specific normalization, the CTABGAN model shows a better performance in detecting multi-modal distributions of the diabetic dataset. This is mainly related to the inclusion of the mode vector in the CTABGAN’s conditional vector. Since our WGAN implementation lacks this specific normalization technique, it struggles to detect complex multi-modal distributions and face mode-collapse. Regarding categorical features of the diabetes dataset, CTGAN outperforms CTABGAN and TGAN models by 2% and 4%, respectively. Both CTGAN and CTABGAN models benefit from a conditional architecture and the training-by-sampling approach to synthesize categorical columns. However, it seems that the extended version of the conditional vector in the CTABGAN model slightly decreases its performance in generating imbalanced categorical columns, resulting in the CTGAN being the best model. Although WGAN is not implemented to deal with the class imbalance issue, it yields an acceptable C-S score of 0.78. this demonstrates that there are only a few imbalanced categorical columns in the diabetic dataset.

	Diabetes		MIMIC III		Thyroid		Epileptic	
	K-S	Chi-S	K-S	Chi-S	K-S	Chi-S	K-S	Chi-S
WGAN	0.75	0.78	0.46	0.94	0.61	0.54	0.94	NaN
TGAN	0.87	0.95	0.72	0.97	0.82	0.87	0.69	NaN
CTGAN	0.87	0.99	0.84	0.99	0.89	0.98	0.84	NaN
CTABGAN	0.90	0.97	0.96	0.97	0.91	0.96	0.90	NaN

Table 5.2: The average Kolmogorov–Smirnov tests and Chi-squared tests between the columns of the original and synthetic datasets.

In the MIMIC III dataset, the CTABGAN model has the best performance in generating numerical columns with a K-S score of 0.96, followed by CTGAN and TGAN with 0.84 and 0.72, respectively. CTABGAN outperforms other models significantly due to its specific design to efficiently synthesize long-tailed numerical distributions using an extra logarithmic transformation pre-processing step. Moreover, it can be observed that the WGAN model completely fails in generating this kind of distribution. Regarding categorical columns, since the dataset does not include imbalanced categorical features, all models demonstrate an outstanding result, with the CTGAN being the best performing and other models following it.

We observe a similar pattern for numerical features in the thyroid dataset compared to the MIMIC III. The long-tailed numerical columns result in the CTABGAN being the best model with a score of 0.91 and CTGAN, TGAN, and WGAN in the next places. A similar pattern compared to Diabetes dataset can be observed for categorical variables. The CTGAN model with a C-S test score of 0.98 outperforms CTABGAN, TGAN, and WGAN by 2%, 11%, and 44%, respectively. However, in contrast to the Diabetes dataset, the WGAN model performs significantly worse due to the plethora of imbalanced categorical features in the Thyroid dataset.

In the Epileptic dataset, we observe a completely different pattern compared to the previous datasets. Interestingly, the WGAN is the best performing model with a score of 0.94. the CTABGAN is slightly worse with a score of 0.90, followed by CTGAN and TGAN with 0.84 and 0.69, respectively. It can be observed that the Mode-specific normalization technique implemented in the CTGAN, TGAN, and CTABGAN models is well-suited for continuous numerical columns with complex distributions. In contrast, the WGAN model delivers its best performance in the smaller datasets with the discrete numerical features. Moreover, since there is no categorical feature in the Epileptic dataset, no chi-squared tests are conducted.

In brief, the CTABGAN model outperforms others in preserving the marginal distribution of numerical columns in the Diabetes, MIMIC III, and Thyroid datasets.

This is due to using a specific normalization and a modified condition vector in its conditional architecture. However, the performance of the CTABGAN in the numerical columns of the Epileptic dataset is slightly decreased due to the existing incompatibility between the wide-ranging integer columns and the CTABGAN’s normalization. The CTGAN is the best-performing model regarding the categorical variables due to its conditional generator and the training-by-sampling approach in the Diabetes, MIMIC III, and Thyroid datasets.

5.2 Machine Learning-based Evaluation

To compare the inferential ability of the original and synthetic datasets, we train a set of predictive models on both the real and fake datasets and compare their predictive capability using the real data. Since all the chosen datasets include a categorical target column, we use the Macro-F1 score to evaluate the predictive capabilities of the models. Macro-F1 is used instead of accuracy due to the imbalanced nature of many categorical features across the investigated datasets. The goal is to verify if the same insights are derived from real and fake datasets when trained on an equally tuned machine learning model, not picking the best classifier. Thus, we exclude hyper-parameter tuning for each predictive model and compare the SDG models based on the average Macro-F1 scores of the classifiers. Besides the inferential comparison, we train Logistic Regression and SVM classifiers on the labelled original and fake datasets to evaluate whether they are distinguishable or not. We compute the normalized AUROC score with one indicating the real and synthetic data are inseparable and zero otherwise. Tables 5.3 and 5.4 show the absolute difference in Macro-F1 scores of the Decision Tree, Random Forest, Logistic Regression and Multi-layer Perceptron classifiers trained on the original and synthetic datasets. WG, TG, CT and CTAB stand for WGAN, TGAN, CTGAN and CTABGAN respectively and the normalized AUROC scores are reported in the last two rows of both tables. A synthetic dataset with a low Macro-F1 difference and high machine learning detection score (normalized AUROC) is considered ideal.

Looking at the tables, if we average the Macro-F1 differences across all four classifiers in the Diabetes dataset, we find that the CTABGAN model has the lowest average score of 0.015, followed by CTGAN and TGAN models with average scores of 0.041 and 0.068. This pattern is repeated in the MIMIC III and Thyroid datasets, with the CTABGAN model outperforming others in terms of the difference in Macro-F1 classification scores followed by the CTGAN, TGAN, and WGAN. The reason why CTABGAN is the best performing model in these datasets is due to using a modified conditional GAN architecture and an additional information loss term in the optimization process. Although the Diabetes, MIMIC III, and Thyroid datasets follow a

	Diabetes				MIMIC III			
	WG	TG	CT	CTAB	WG	TG	CT	CTAB
Δ F1-DT	0.083	0.056	0.030	0.016	0.189	0.068	0.052	0.031
Δ F1-RF	0.122	0.082	0.044	0.023	0.120	0.030	0.020	0.010
Δ F1-LR	0.129	0.050	0.046	0.009	0.094	0.001	0.015	0.007
Δ F1-MLP	0.143	0.087	0.044	0.013	0.015	0.035	0.039	0.028
Δ F1-average	0.119	0.068	0.041	0.015	0.104	0.033	0.031	0.019
LR-D	0.330	0.520	0.540	0.700	0.220	0.620	0.730	0.790
SVM-D	0.110	0.290	0.330	0.560	0.110	0.390	0.480	0.540

Table 5.3: The difference of the Macro-F1 classification scores and the machine learning detection scores in the Diabetes and MIMIC III datasets.

similar pattern regarding the average Macro-F1 scores, there is a large gap between the CTABGAN and CTGAN in the Diabetes dataset compared to the ones in the other two datasets. This can be related to the multi-modal nature of the numerical columns in the Diabetes dataset and how the CTABGAN model successfully generates this type of numerical distribution, benefiting from an extended condition vector in its architecture.

However, in the Epileptic dataset, the WGAN outperforms other models regarding the average Macro-F1 differences. WGAN is the best performing model with an average score of 0.085, followed by CTABGAN, TGAN, and CTGAN, with average scores of 0.18, 0.21, and 0.23, respectively. Similar to our interpretation in Section 5.1.3 for the Epileptic case, we find that although the Mode-specific normalization approach in the CTGAN, TGAN, and CTABGAN is well-suited for numerical columns with complex distributions, it may prevent the model from reaching an ideal optimum in the smaller datasets with the discrete numerical variables (integers).

The Logistic Regression and SVM classifiers’ normalized AUROC scores follow the same pattern as the average Macro-F1 differences. Due to using an auxiliary classifier in the CTABGAN’s architecture and utilizing a classification loss term in its optimization process, it is much harder to distinguish the synthetic data generated from the CTABGAN model from the original one. In other words, in the Diabetes, MIMIC III, and Thyroid datasets, the normalized classification AUROC scores of the CTABGAN model outperform the Scores resulting from the other models. Similarly, in the Epileptic case, the WGAN’s normalized AUROC score exceeds the scores of the other models, followed by CTABGAN, CTGAN, and TGAN.

Briefly, the CTABGAN model shows the best performance when comparing the machine learning cross-testing scores in the Diabetes, MIMIC III, and Thyroid datasets.

	Thyroid				Epileptic			
	WG	TG	CT	CTAB	WG	TG	CT	CTAB
$\Delta F1$ -DT	0.300	0.240	0.200	0.100	0.070	0.210	0.210	0.180
$\Delta F1$ -RF	0.260	0.220	0.190	0.100	0.050	0.280	0.250	0.230
$\Delta F1$ -LR	0.140	0.090	0.050	0.050	0.010	0.010	0.080	0.030
$\Delta F1$ -MLP	0.240	0.170	0.120	0.080	0.210	0.350	0.390	0.280
$\Delta F1$ -average	0.230	0.180	0.140	0.080	0.085	0.210	0.230	0.180
LR-D	0.530	0.700	0.700	0.780	0.770	0.330	0.450	0.730
SVM-D	0.380	0.600	0.530	0.620	0.620	0.250	0.380	0.590

Table 5.4: The difference of the Macro-F1 classification scores and the machine learning detection scores in the Thyroid and Epileptic datasets.

The CTABGAN model outperforms the others significantly in these datasets due to its modified conditional GAN architecture and an additional information loss term in its optimization process. However, in the Epileptic dataset, the WGAN model outperforms others due to the incompatibility between the TGAN, CTGAN, and CTABGAN normalization techniques and the wide-ranging integer columns of the Epileptic dataset. Moreover, the normalized AUROC scores of the Logistic Regression and SVM trained on the real and fake data demonstrate the exact pattern as the ML cross-testing scores, with the CTABGAN model being the best-performing model in generating synthetic records indistinguishable from the original ones.

5.3 Preserved Privacy

Finally, we must evaluate the proposed SDG models regarding their potential to preserve the privacy of sensitive data. Specifically, this evaluation category is highlighted in the healthcare domain, where patients share their sensitive and private information. Suppose a patient’s confidential data is to be re-identified by accessing the synthetic data. In that case, the patient’s sensitive information is undoubtedly leaked in the synthetic dataset, and the SDG model simply replicates the original records when generating new ones. Thus, as with other related publications, we utilize a distance-based metric to give an overview of the preserved privacy in the generated datasets. We compute the Euclidean distance between each synthetic record and its closest counterpart in the original dataset. Consequently, the distribution of pairwise distances between each synthetic record and its nearest original neighbor is achieved. It is preferred that the resulting distribution has a large mean and small standard deviation when evaluating the models through the lens of privacy. However, this conflicts with the evaluation of synthetic data from the ML

utility perspective (discussed in section 5.2). Generally, if the distance between the original and fake records is too large, the generated data has a poor quality. In other words, these evaluation categories (privacy and ML-based utilities) are inversely proportional. Thus, for a fair comparison, we should consider privacy and ML utility evaluations simultaneously.

Model	Diabetes	MIMIC III	Thyroid	Epileptic
WGAN	3.10 ± 0.46	1.37 ± 1.14	1.88 ± 0.97	7.55 ± 8.45
TGAN	2.69 ± 0.61	0.93 ± 0.80	1.39 ± 1.05	7.81 ± 9.26
CTGAN	2.71 ± 0.64	0.97 ± 0.90	1.34 ± 1.08	9.10 ± 9.18
CTABGAN	3.02 ± 0.46	1.11 ± 0.84	1.76 ± 1.06	8.08 ± 9.27

Table 5.5: The distribution of Euclidean distances between synthetic records and their closest original counterparts (DCR distributions). The format is *mean* \pm *std*.

Table 5.5 shows the mean and standard deviation of the DCR distributions. In the Diabetes, MIMIC III, and Thyroid datasets, the WGAN model maintains the largest distance between the original and synthetic data (lowest privacy risk). This verifies the results in Section 5.2, as the WGAN model was the worst-performing regarding ML utilities. Interestingly, in contrast to the results for ML utilities, we observe that the CTABGAN is the second best-performing model with a mean of 3.02 and a standard deviation of 0.46. although the CTABGAN model outperforms the CTGAN and TGAN models in these three datasets, it preserves more degree of privacy when generating synthetic records.

In the epileptic dataset, the CTGAN and WGAN models are the best and worst SDG models regarding privacy preservation, with the means of 9.10 and 7.55, respectively. This verifies the results in ML utilities as the WGAN is the best-performing and CTGAN is the worst-performing one in the Epileptic case. Moreover, the CTABGAN model shows an impressive performance in preserving privacy and the ML utility metrics as it is the second best-performing model in both evaluation categories.

In a nutshell, if we evaluate the distribution of each synthetic record to its closest original counterpart under the equivalent ML-based scores, the model with the larger mean and smaller standard deviation is the best-performing model in preserving privacy. Thus, considering the mean and standard deviations of the DCR distributions and ML-based scores simultaneously, it can be observed that the CTABGAN model outperforms others in preserving the privacy of the original records in all datasets.

5.4 Summary of the Overall Results

In this chapter, various evaluation metrics are studied to shed light on the strengths and weaknesses of each tabular SDG model in different healthcare datasets. Specifically, we evaluate the models in terms of statistical resemblance, ML-based evaluation scores, and privacy metrics in the Diabetes, MIMIC III, Thyroid, and Epileptic datasets. One important observation is that the evaluation results for each model in the Diabetes, MIMIC III, and Thyroid datasets follow similar patterns, as opposed to the Epileptic dataset with a different one. This is directly due to the size and the nature of the investigated datasets. Although we deal with complex column distributions in the Diabetes, MIMIC III, and Thyroid datasets, the Epileptic Seizure Recognition dataset consists of wide-ranging discrete numerical variables. This indicates that there is room for the state-of-the-art models to improve especially when generating integer columns.

The CTABGAN model outperforms other proposed models in the Diabetes, MIMIC III, and Thyroid datasets regarding the statistical resemblance and ML-based evaluation metrics, except for the correlation difference. Although the CTABGAN model advances beyond the TGAN, CTGAN, and WGAN implementations in statistical similarity and model compatibility metrics due to its mode-specific normalization, the novel modification of its conditional generator, and the use of classification and information losses in its training process, it is observed that the sequential generation of each record in the TGAN’s generator (LSTM network) is a better approach in preserving the correlation between variables.

However, in the Epileptic dataset, the WGAN outperforms other advanced SDG models in all statistical resemblance and ML-based evaluation metrics. As discussed in the previous sections, it can be concluded that the mode-specific normalization technique in the TGAN, CTGAN, and CTABGAN models is only well-suited for continuous numerical columns with complex distributions. However, this normalization technique significantly reduces the performance of models in generating wide-ranging integer columns.

Finally, we compute the Euclidean distances between each synthetic record and its closest counterpart and compare the means and standard deviations of the resulting distributions (DCR distributions) as a measure of preserved privacy. Since his notion of privacy (distance-based privacy metric) is inversely proportional to the ML-based evaluation scores, we evaluate them simultaneously to find the best-performing model in model compatibility and preserving privacy. Accordingly, it is observed that the CTABGAN model outperforms others in maintaining the privacy of the original records in all datasets.

Chapter 6

Discussion

This chapter presents an in-depth discussion of the results obtained in Chapter 5, the efficacy of the proposed tabular SDG models in different column types and the thesis limitations.

6.1 In-depth analysis of Distributions

In this section, we conduct an in-depth analysis of the marginal distributions to compare the performance of each SDG model when generating columns with specific distributions. We analyze the efficacy of the models in generating multi-modal, discrete numerical, long-tailed, and imbalanced categorical columns in the following sections.

6.1.1 Multi-modal Numerical Columns

Figure 6.1 compares the marginal distributions (top row) and the cumulative distributions (bottom row) of the `diag_2` numerical column in the Diabetes dataset. Looking at the original data, it is clear that the marginal distribution has a dominant peak at 450 and multiple lower peaks around it. The WGAN implementation lacks any specific normalization to detect various modes in the numerical features and generates a simple normal distribution around the dominant peak (it faces mode-collapse). Although the Wasserstein GAN loss function was introduced to circumvent the mode-collapse issue, we observe that it is not applicable to detect complex multi-modal distributions as in our case.

On the other hand, the other three models utilize the mode-specific normalization technique for dealing with multi-modal continuous columns and clearly excel

WGAN in capturing the modes of the `diag_2` column. Comparing the cumulative distributions of the TGAN, CTGAN, and CTABGAN models demonstrate that the CTABGAN architecture outperforms the other two in generating skewed multi-modal numerical columns. This is due to the modification of the conditional vector compared to the CTGAN model. Besides the one-hot encoded representations for categorical columns, the extended conditional vector in CTABGAN includes the mode of the numerical columns, increasing its performance in capturing modes with less weight.

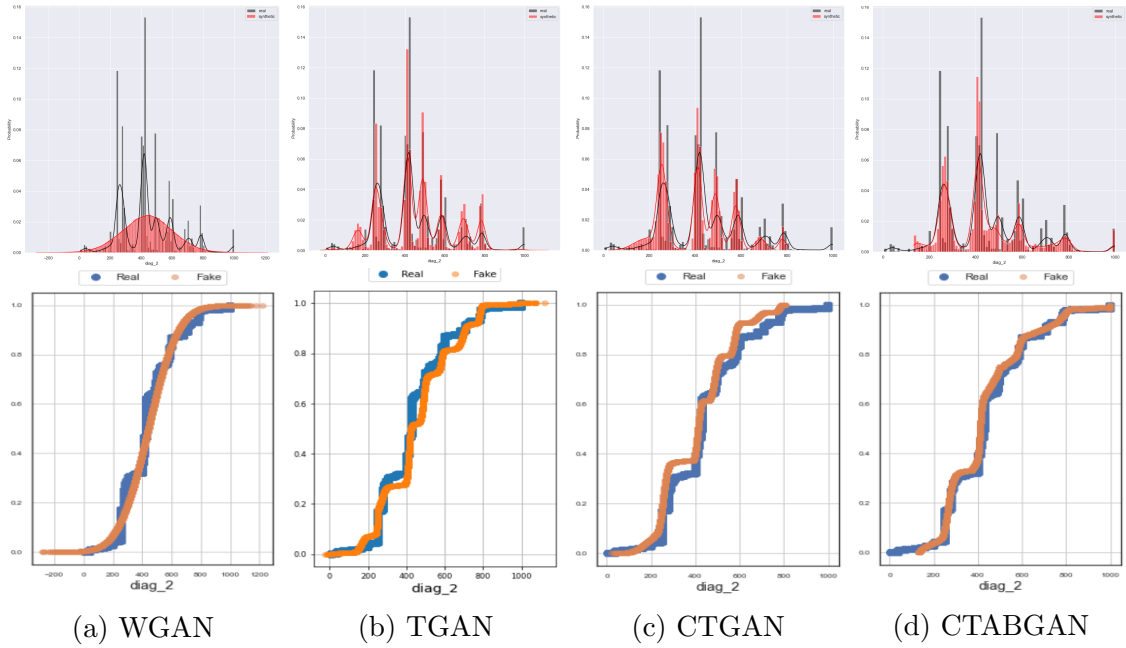


Figure 6.1: The marginal and cumulative probability distributions for the multi-modal `diag_2` column in the Diabetes dataset.

6.1.2 Discrete Numerical Columns

One of the few areas where there is room for further improvement is the generation of discrete numerical columns, as none of the proposed models makes any distinction between the continuous and discrete numerical features. Through our experimentation, we observe that for small-range discrete numerical columns, the original and synthetic distribution tend to resemble perfectly. However, the models' performances significantly drop when generating integer columns with a wide range of values. Figure 6.2 shows the marginal probability distributions of two discrete numerical columns in the Diabetes dataset. One column has a small range of integer values and the other has a wide range.

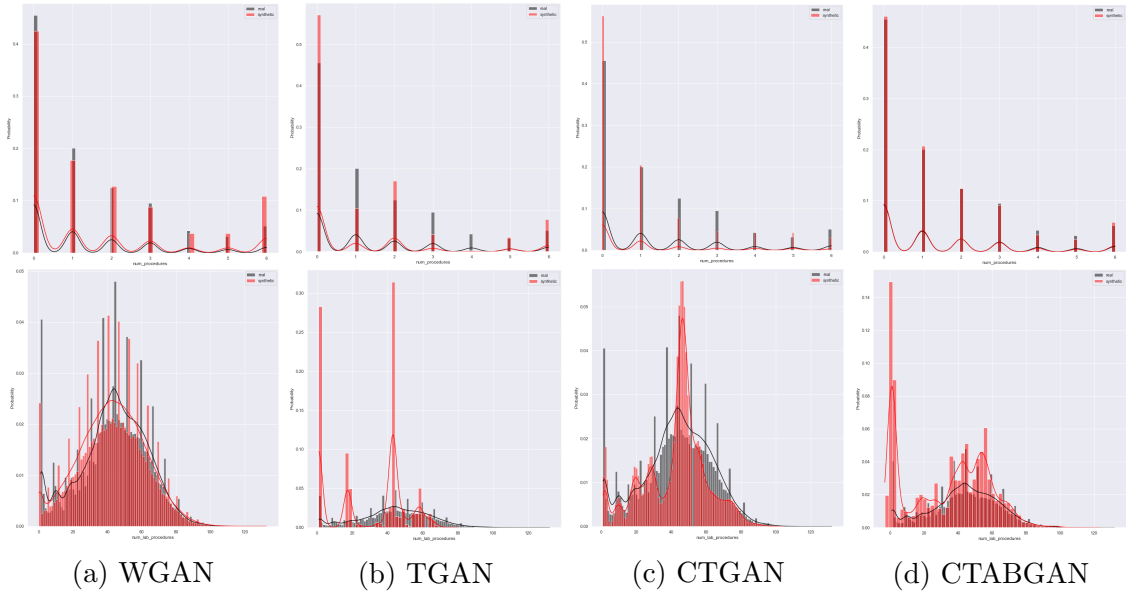


Figure 6.2: The marginal distributions of two integer columns with a small range (upper row) and a large range of values (lower row) in the Diabetes dataset.

For the integer column with a small range of values, all the models have acceptable performance, with the CTABGAN and WGAN being the best-performing ones. However, we observe a completely different pattern for the wide-ranging integer column. While the marginal distribution of the original data resembles a simple Gaussian distribution, the probability distributions resulting from the TGAN, CTGAN, and CTABGAN consist of several modes. This occurs due to the mode-specific normalization implemented in the mentioned SDG models. Although this technique is well-suited for continuous columns with complex distributions, it may prevent the model from reaching an ideal optimum for the discrete numerical variables.

On the other hand, the WGAN is the best performing model when generating wide-ranging integer variables. This pattern is repeated in the Epileptic dataset consisting of 178 integer columns with wide ranges. As observed in Sections 5.1.3 and 5.2, the WGAN outperforms other models in terms of the ML utilities and statistical similarities due to the same reasons.

6.1.3 Long-tailed Numerical Columns

In this section, we compare the ability of the proposed models to generate numerical columns with long-tailed distributions. In this type of continuous distribution, most of the observations happen near the head of the distribution, while there exist many cases with extreme values.

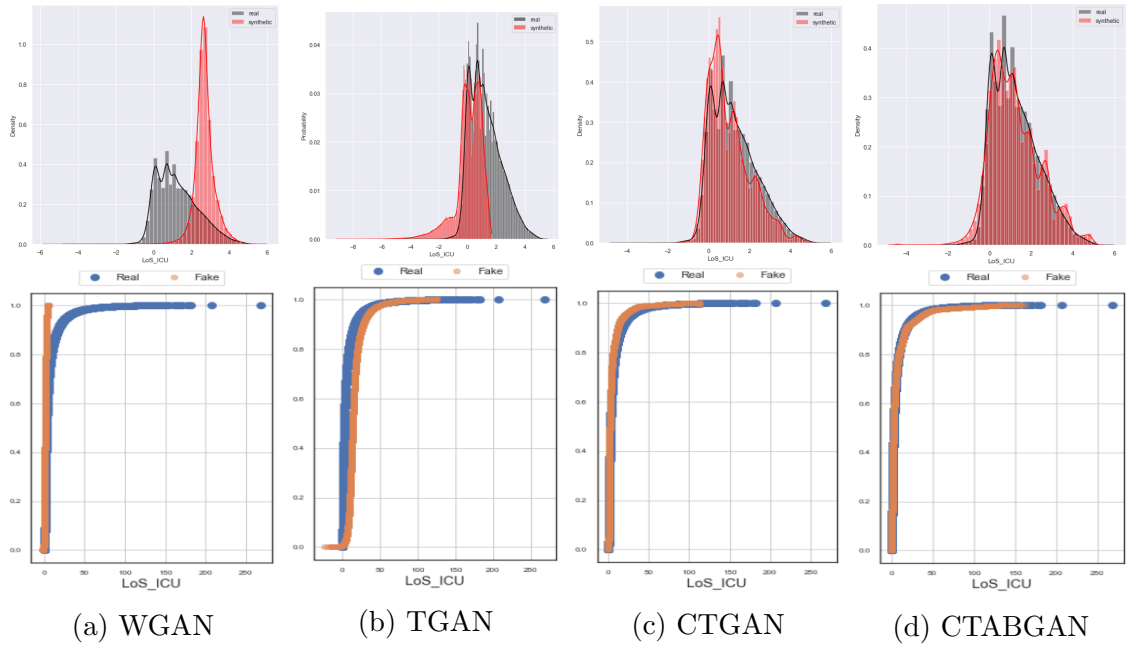


Figure 6.3: The marginal and cumulative probability distributions for the long-tailed *los_ICU* column in the MIMIC III dataset.

Figure 6.3 illustrates the log-transformed marginal probability distributions (top row) and the cumulative probability distributions (bottom row) of a numerical column with a long tail. The chosen column represents the patients’ length of stay in the ICU in the MIMIC III dataset. It is clear that the distribution of the values continue until 250, although most of the observations lie in the initial part ranging between 0 to 50. Based on the CDF plots, it is observed that the CTABGAN model outperforms the CTGAN and TGAN in generating long-tailed distributions, while the WGAN model completely fails. This is directly related to the log-transformation technique implemented in the CTABGAN pre-processing.

6.1.4 Imbalanced Categorical Columns

Lastly, we analyze the performance of the tabular SDG models in terms of generating highly imbalanced categorical features, in which the major class includes more than 90% of the data. Since a minor category slightly impacts the joint distribution of the original data, it is often missed when generating synthetic data due to insufficient model training. Figure 6.4 illustrates the probability mass functions (top row) and the cumulative probability distributions (bottom row) of an imbalanced column in the Thyroid dataset.

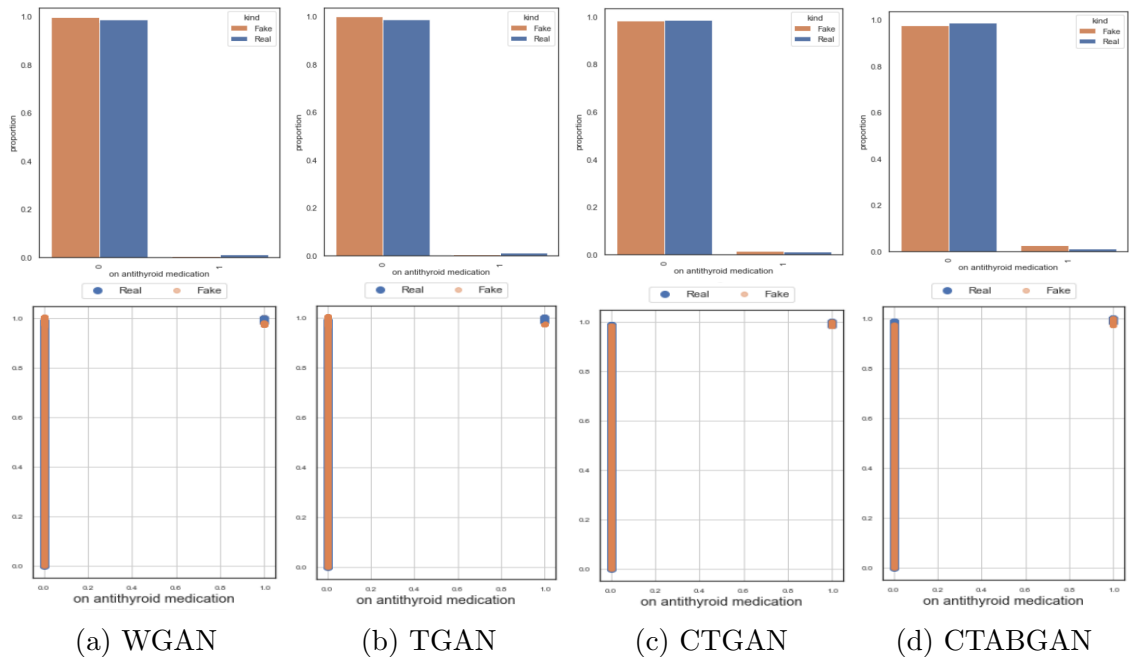


Figure 6.4: The probability mass functions and cumulative probability distributions of the *antithyroid medication* column in the Thyroid dataset.

As observed, the CTABGAN and CTGAN models outperform the other two when generating the minor classes. This is due to the conditional generator and training-by-sampling technique employed in these two models to handle class-imbalanced features. Comparing the distributions of the original and synthetic data reveals that the minor categories are generated more frequently than those in the actual data. This weakness is directly related to how the training-by-sampling technique gives importance to the major and minor classes based on the logarithm of each type. Moreover, as discussed in Section 5.1.3, the overall performance of the CTGAN model in generating categorical columns, whether they are imbalanced or not, slightly outperforms the CTABGAN model. Although the extended conditional vector in the CTABGAN model improves the performance in generating numerical columns with complex distributions, it slightly decreases the performance when generating categorical ones.

In a nutshell, the overall comparison of the proposed SDG models demonstrates that the CTABGAN is the best-performing model in generating continuous distributions. Specifically, CTABGAN outperforms others in generating long-tailed numerical and skewed multi-modal distributions due to using a logarithmic transformation pre-processing and a modified conditional generator in its architecture. Although both CTGAN and CTABGAN use a conditional generator and the training-by-sampling

technique to handle generating imbalanced categorical features, it is observed that the CTGAN model slightly outperforms the CTABGAN architecture when dealing with highly imbalanced categorical variables. This is due to the modification of the CTABGAN’s conditional generator to capture skewed multi-modal distributions more effectively compared to the CTGAN model. Moreover, CTABGAN, CTGAN, and even TGAN architectures show relatively poor performances in generating wide-ranging, discrete numerical distributions. This is due to the mode-specific normalization implemented in these models as it prevents them from reaching an ideal optimum for the discrete numerical distribution, especially the wide-ranging ones. This shows that there is still room for these state-of-the-art SDG models to improve.

6.2 Limitations

Although we aim to conduct a detailed comparison between the state-of-the-art GAN-based SDG models in the healthcare use cases, there are some limitations due to the insufficient time and computational power. This section presents some of these limitations.

6.2.1 Data

In Deep Learning methods, especially GAN-based ones, we need a large-sized dataset to train the models. In other words, to generate synthetic data with high quality and diversity, we need a comprehensive training dataset consisting of thousands of records so the generative models can effectively learn the underlying distributions of the columns and the relationships between them. Getting access to such datasets is extremely cumbersome in the healthcare domain due to the confidentiality of the data. In this thesis, we conduct experiments on a limited number of publicly available medical datasets we could gain access to. However, various medical datasets are only accessible to healthcare professionals or medical practitioners. Analyzing these datasets can reveal more exciting properties about the tabular SDG models.

6.2.2 Hyper-parameter Selection

Due to the limitation of time and computational power, we are unable to conduct hyper-parameter tuning for each SDG model in each medical dataset. Our few experiments indicate that the generated tabular data is not highly sensitive to hyper-parameter optimization in ML-based applications. Consequently, we use the default hyper-parameters of the SDG models in many experiments and leave a wide-ranging hyper-parameter search for future studies.

6.2.3 Convergence

To conduct a fair comparison, we train the proposed generative models for 300 epochs in the medical datasets, including more than 40,000 records and 1,000 epochs for datasets with less than 10,000 records. We only examine the synthetic data through the lens of their practical use cases and compare the tabular SDG models based on the proposed external evaluation metrics. Although it is common in the GAN-based architectures to tune the training process based on the generator and discriminators' loss values, using Wasserstein loss often prevents the model to reach Nash equilibrium. Furthermore, our limited trials demonstrate that there are cases where the adversarial components have small losses, and the evaluation metrics are still being improved. In other words, there is no direct relationship between the loss values and evaluation metrics. Thus, we focus on comparing the performance of the models trained on a specific number of epochs based on the number of records in each dataset. However, it would be interesting to incorporate the ML-based evaluation scores in the training process and implement an early stopping approach to improve the stability by evaluating the model continuously.

Chapter 7

Conclusions & Recommendations

This chapter briefly summarizes what has been done through the thesis. First, we conclude the thesis with our findings and final thoughts in Section 7.1. Then, in Section 7.2, we recommend some possible suggestions for future studies.

7.1 Conclusions

Due to the rapid development of the healthcare industry, there is a growing need for large amounts of reliable data. Specifically, tabular data plays a crucial role in the modern advancements of healthcare data-driven applications. However, strict data protection regulations are introduced to safeguard individuals' sensitive information, especially in the health sector. These privacy-protection laws severely slow the pace of the development of healthcare applications. To circumvent these obstacles, synthetic data is an ideal alternative to be used instead of the original data without compromising privacy.

This thesis aims to evaluate the strengths and weaknesses of several state-of-the-art GAN-based synthetic data generation models in tabular medical datasets. First, we begin our thesis with a comprehensive literature study, where we conduct a systematic review of the proposed GAN-based SDG models and get insights from related research papers. After carrying out the literature study phase, TGAN, CTGAN, CTABGAN, and WGAN-GP are the selected SDG models to be investigated in detail. The WGAN-GP model is implemented as a baseline model to be compared against others. Then, we conduct an extensive survey on the existing medical tabular datasets and their applicability to be used in the SDG process. To determine the

efficacy of each model in generating columns with different feature distributions and data types, we select Diabetes, MIMIC III, Thyroid, and Epileptic datasets. Later, a complete pre-processing pipeline is implemented for each tabular dataset to make them ready and feasible for the SDG task. The next step involves defining a proper evaluation framework to assess the quality of the generated data and the reliability of each SDG model in preserving the privacy of individuals. Each synthetic dataset is evaluated based on statistical similarity metrics, ML-based evaluation scores, and distance-based privacy metrics. These evaluation metrics empirically highlight the strengths and weaknesses of each tabular SDG model.

Our experiments indicate that the synthetic tabular data is not highly sensitive to hyper-parameter tuning in ML-based applications. Thus, the hyper-parameters of the selected models are chosen based on their recommended values in the original papers. Furthermore, the models are trained for 300 epochs in the Diabetes and MIMIC III and 1,000 epochs in the Thyroid and Epileptic datasets. Due to the limited number of training records in the Thyroid and Epileptic datasets, a much larger number of epochs is required for the models to converge.

The experimental results produced in our thesis suggest that the state-of-the-art SDG models can deliver outstanding performances in preserving statistical characteristics, model compatibility, and integrity of the original data in the generation process. Specifically, it is demonstrated that the CTABGAN model outperforms TGAN, CTGAN and WGAN in the statistical similarity metrics, ML-based evaluation scores, and distance-based privacy metrics. This excellent performance is due to the CTABGAN’s mode-specific normalization, the novel modification of its conditional generator compared to the CTGAN model, and the use of classification and information losses in its training process. However, it has been discovered that the model has significant flaws in preserving the correlations and generating the discrete numerical (integer) columns containing a wide range of values. This indicates that there is still room for further improvements in designing a perfect architecture for generating synthetic tabular data.

To conclude, the obtained visual and quantitative results in our experiments support the primary objectives of the thesis, demonstrating that synthetic healthcare data can be a reliable substitute for the original data and the proposed methodology can be employed to generate tabular healthcare datasets that are statistically similar to the original ones without leaking individuals’ sensitive information. Furthermore, the proposed workflow eliminates the need for traditional anonymization and obfuscation techniques which are too risky and negatively impact the data utilities.

Lastly, the main findings and contributions of the thesis are summarized in a research article that will be submitted to the journal of Artificial Intelligence for Medicine.

7.2 Future Work

Although our experiments demonstrate the promising performance of the proposed SDG models in generating healthcare datasets, there is still room for the SDG models to improve further and address the existing limitations. In the following, we present multiple research directions and suggest some personal recommendations that can be explored as the future work of the current thesis.

- The proposed SDG models are limited to only generating numerical and categorical features. An exciting research direction is finding a way to include other data types like ordinal values, dates, free texts, time series, bounded numerical data, etc., in the generation process.
- The proper evaluation metrics for the tabular GAN-based architectures are still hot topics to investigate. There are various approaches for evaluating synthetic data we do not consider due to time limitations. For instance, Differential Privacy is another guarantee introduced to protect individuals' privacy in the generation process. Thus, implementing a differential privacy supporting version of the proposed models can be addressed in the future. Furthermore, clinical evaluations can be conducted in the last step to see if the generated records are meaningful in clinical settings.
- We only focus on comparing the performance of the SDG models trained on a specific number of epochs based on the number of records in each dataset.

However, it would be interesting to incorporate the ML-based evaluation scores in the training process and implement an early stopping approach to improve the stability and training convergence by evaluating the model continuously.

- Further developments in the models' architectures can potentially improve the performance of synthetic data generation. For instance, the CTABGAN model shows slightly poor performance in preserving the column correlations. We can combine the TGAN and CTABGAN generator structures and use the LSTM network in the CTABGAN's conditional generator to see if it significantly affects the correlations. Furthermore, there is much room for the proposed SDG models to improve training convergence in small-sized datasets and generate discrete numerical (integer) columns.
- In this thesis, we only investigate the strengths and weaknesses of the GAN-based SDG models in generating healthcare tabular datasets. However, it would be interesting to test other generative models like Variational Auto-encoders, Gaussian Copula, Bayesian Networks, etc., in the medical use cases.

Bibliography

- [1] Martin Abadi et al. “Deep learning with differential privacy”. In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.
- [2] Ralph G Andrzejak et al. “Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state”. In: *Physical Review E* 64.6 (2001), p. 061907.
- [3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein generative adversarial networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.
- [4] Mrinal Kanti Baowaly et al. “Synthesizing electronic health records using improved generative adversarial networks”. In: *Journal of the American Medical Informatics Association* 26.3 (2019), pp. 228–241.
- [5] Bauke Brenninkmeijer et al. “On the generation and evaluation of tabular data using GANs”. PhD thesis. Radboud University Nijmegen, The Netherlands, 2019.
- [6] Jason Brownlee. *Generative adversarial networks with python: deep learning generative models for image synthesis and image translation*. Machine Learning Mastery, 2019.
- [7] Jason Brownlee. *How to Choose an Activation Function for Deep Learning*. Ed. by Machine Learning Mastery. URL: <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>.
- [8] Ramiro Camino, Christian Hammerschmidt, and Radu State. “Generating multi-categorical samples with generative adversarial networks”. In: *arXiv preprint arXiv:1807.01202* (2018).
- [9] Bipartisan Policy Center, ed. *AI and Ethics*. URL: <https://bipartisanpolicy.org/report/ai-ethics/>.
- [10] Tong Che et al. “Mode regularized generative adversarial networks”. In: *arXiv preprint arXiv:1612.02136* (2016).

- [11] Edward Choi et al. “Generating multi-label discrete patient records using generative adversarial networks”. In: *Machine learning for healthcare conference*. PMLR. 2017, pp. 286–305.
- [12] João Coutinho-Almeida, Pedro Pereira Rodrigues, and Ricardo João Cruz-Correia. “GANs for Tabular Healthcare Data Generation: A Review on Utility and Privacy”. In: *International Conference on Discovery Science*. Springer. 2021, pp. 282–291.
- [13] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: <http://archive.ics.uci.edu/ml>.
- [14] Cynthia Dwork. “Differential privacy: A survey of results”. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.
- [15] Cynthia Dwork, Aaron Roth, et al. “The algorithmic foundations of differential privacy.” In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2014), pp. 211–407.
- [16] Khaled El Emam et al. “A systematic review of re-identification attacks on health data”. In: *PloS one* 6.12 (2011), e28071.
- [17] Sara Gerke, Timo Minssen, and Glenn Cohen. “Ethical and legal challenges of artificial intelligence-driven healthcare”. In: *Artificial intelligence in healthcare*. Elsevier, 2020, pp. 295–336.
- [18] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* ” O’Reilly Media, Inc.”, 2019.
- [19] Ary L Goldberger et al. “PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals”. In: *circulation* 101.23 (2000), e215–e220.
- [20] Andre Goncalves et al. “Generation and evaluation of synthetic patient data”. In: *BMC medical research methodology* 20.1 (2020), pp. 1–40.
- [21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [22] Ian Goodfellow et al. “Generative adversarial nets”. In: *Advances in neural information processing systems* 27 (2014).
- [23] Ishaan Gulrajani et al. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems* 30 (2017).
- [24] Marte T Gustavsen et al. “Tacrolimus area under the concentration versus time curve monitoring, using home-based volumetric absorptive capillary microsampling”. In: *Therapeutic Drug Monitoring* 42.3 (2020), pp. 407–414. DOI: 10.1097/FTD.0000000000000697.
- [25] Marte Theie Gustavsen et al. “Fasting Status and Circadian Variation Must be Considered When Performing AUC-based Therapeutic Drug Monitoring

- of Tacrolimus in Renal Transplant Recipients”. In: *Clinical and translational science* 13.6 (2020), pp. 1327–1335. DOI: 10.1111/cts.12833.
- [26] Jamie Hayes et al. “Logan: Membership inference attacks against generative models”. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)*. Vol. 2019. 1. De Gruyter. 2019, pp. 133–152.
- [27] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5 (1989), pp. 359–366.
- [28] Jonathan Hui. *GAN - Wasserstein GAN and WGAN-GP*. Ed. by Medium. URL: <https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490>.
- [29] Jonathan Hui. *GAN - Why it is so hard to train Generative Adversarial Networks!* Ed. by Medium. URL: <https://jonathan-hui.medium.com/gan-why-it-is-so-hard-to-train-generative-adversarial-networks-819a86b3750b#4987>.
- [30] Jonathan Hui. *What is Generative Adversarial Networks GAN?* Ed. by Medium. URL: <https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09>.
- [31] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.
- [32] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical reparameterization with gumbel-softmax”. In: *arXiv preprint arXiv:1611.01144* (2016).
- [33] Alistair EW Johnson et al. “MIMIC-III, a freely accessible critical care database”. In: *Scientific data* 3.1 (2016), pp. 1–9.
- [34] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. “PATE-GAN: Generating synthetic data with differential privacy guarantees”. In: *International conference on learning representations*. 2018.
- [35] Saeed Karimi-Bidhendi et al. “Fully-automated deep-learning segmentation of pediatric cardiovascular magnetic resonance of patients with complex congenital heart diseases”. In: *Journal of cardiovascular magnetic resonance* 22.1 (2020), pp. 1–24.
- [36] Anton Karlsson and Torbjörn Sjöberg. *Synthesis of tabular financial data using generative adversarial networks*. 2020.
- [37] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [38] Frederik Kratzert. *Understanding the backward pass through Batch Normalization Layer*. Ed. by Flair of Machine Learning. URL: <https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html>.

- [39] Aditya Kinar. “Effective and Privacy preserving Tabular Data Synthesizing”. In: *arXiv preprint arXiv:2108.10064* (2021).
- [40] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [41] Zinan Lin et al. “Pacgan: The power of two samples in generative adversarial networks”. In: *Advances in neural information processing systems* 31 (2018).
- [42] Mikael Ljung. *Synthetic Data Generation for the Financial Industry Using Generative Adversarial Networks*. 2021.
- [43] BERNHARD Mehlig. “Machine learning with neural networks”. In: *arXiv preprint arXiv:1901.05639* (2019).
- [44] Luke Metz et al. “Unrolled generative adversarial networks”. In: *arXiv preprint arXiv:1611.02163* (2016).
- [45] Karsten Midtvedt et al. “No change in insulin sensitivity in renal transplant recipients converted from standard to once-daily prolonged release tacrolimus”. In: *Nephrology Dialysis Transplantation* 26.11 (2011), pp. 3767–3772. DOI: 10.1093/ndt/gfr153.
- [46] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).
- [47] Andrew Montanez et al. “SDV: an open source library for synthetic data generation”. PhD thesis. Massachusetts Institute of Technology, 2018.
- [48] Aiden Nibali. *The GAN objective, from practice to theory and back again*. Ed. by Medium. URL: <https://aiden.nibali.org/blog/2016-12-21-gan-objective/#fn:igo>.
- [49] Sofia Nord. *Multivariate Time Series Data Generation using Generative Adversarial Networks: Generating Realistic Sensor Time Series Data of Vehicles with an Abnormal Behaviour using TimeGAN*. 2021.
- [50] Noseong Park et al. “Data synthesis based on generative adversarial networks”. In: *arXiv preprint arXiv:1806.03384* (2018).
- [51] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [52] Ben Poole et al. “Improved generator objectives for gans”. In: *arXiv preprint arXiv:1612.02780* (2016).
- [53] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434* (2015).
- [54] Sina Rashidian et al. “SMOOTH-GAN: towards sharp and smooth synthetic ehr data generation”. In: *International Conference on Artificial Intelligence in Medicine*. Springer. 2020, pp. 37–48.

- [55] Karim Rezaul. *Generative adversarial networks*. Ed. by O'Reilly. URL: <https://www.oreilly.com/library/view/java-deep-learning/9781788997454/60579068-af4b-4bbf-83f1-e988fbe3b226.xhtml>.
- [56] Ida Robertsen et al. "Use of generic tacrolimus in elderly renal transplant recipients: precaution is needed". In: *Transplantation* 99.3 (2015), pp. 528–532. DOI: 10.1097/TP.0000000000000384.
- [57] Tiago Rosa dos Reis. *Measuring the statistical similarity between two samples using Jensen-Shannon and Kullback-Leibler divergences*. Ed. by Medium. URL: <https://medium.com/datalab-log/measuring-the-statistical-similarity-between-two-samples-using-jensen-shannon-and-kullback-leibler-8d05af514b15>.
- [58] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. URL: <https://ruder.io/optimizing-gradient-descent/>.
- [59] Tim Salimans et al. "Improved techniques for training gans". In: *Advances in neural information processing systems* 29 (2016).
- [60] Reza Shokri et al. "Membership inference attacks against machine learning models". In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.
- [61] Joshua Snoke et al. "General and specific utility measures for synthetic data". In: *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 181.3 (2018), pp. 663–688.
- [62] Beata Strack et al. "Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records". In: *BioMed research international* 2014 (2014).
- [63] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *arXiv preprint arXiv:1511.01844* (2015).
- [64] Amber Thomas. *Exploring the Kaggle Data Science Survey*. Ed. by DataCamp. URL: <https://rpubs.com/cliex159/868800>.
- [65] Amirsina Torfi and Edward A Fox. "CorGAN: Correlation-capturing convolutional generative adversarial networks for generating synthetic healthcare records". In: *The Thirty-Third International Flairs Conference*. 2020.
- [66] Amirsina Torfi, Edward A Fox, and Chandan K Reddy. "Differentially private synthetic medical data generation using convolutional gans". In: *Information Sciences* 586 (2022), pp. 485–500.
- [67] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. "Dp-cgan: Differentially private synthetic data and label generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.
- [68] Roberto Vitillo. *Differential privacy for dummies*. URL: <https://robertovitillo.com/differential-privacy-for-dummies/>.

- [69] Wikipedia. *Correlation ratio*. URL: https://en.wikipedia.org/wiki/Correlation_ratio.
- [70] Wikipedia. *Cramér's V coefficient*. URL: https://en.wikipedia.org/wiki/Cram%C3%A9r's_V.
- [71] Wikipedia. *Pearson correlation coefficient*. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.
- [72] Wikipedia. *Uncertainty coefficient*. URL: https://en.wikipedia.org/wiki/Uncertainty_coefficient.
- [73] Liyang Xie et al. “Differentially private generative adversarial network”. In: *arXiv preprint arXiv:1802.06739* (2018).
- [74] Lei Xu and Kalyan Veeramachaneni. “Synthesizing tabular data using generative adversarial networks”. In: *arXiv preprint arXiv:1811.11264* (2018).
- [75] Lei Xu et al. “Modeling tabular data using conditional gan”. In: *Advances in Neural Information Processing Systems* 32 (2019).
- [76] Andrew Yale et al. “Generation and evaluation of privacy preserving synthetic health data”. In: *Neurocomputing* 416 (2020), pp. 244–255.
- [77] Andrew Jonathan Yale. *Privacy Preserving Synthetic Health Data Generation and Evaluation*. Rensselaer Polytechnic Institute, 2020.
- [78] Jinsung Yoon, Lydia N Drumright, and Mihaela Van Der Schaar. “Anonymization through data synthesis using generative adversarial networks (ads-gan)”. In: *IEEE journal of biomedical and health informatics* 24.8 (2020), pp. 2378–2388.
- [79] Zilong Zhao et al. “CTAB-GAN: Effective Table Data Synthesizing”. In: *Asian Conference on Machine Learning*. PMLR. 2021, pp. 97–112.
- [80] Shaked Zychlinski. *The Search for Categorical Correlation*. Ed. by Towards Data Science. URL: <https://towardsdatascience.com/the-search-for-categorical-correlation-a1cf7f1888c9>.

Appendix A

Diabetes

A.1 Basic Similarities

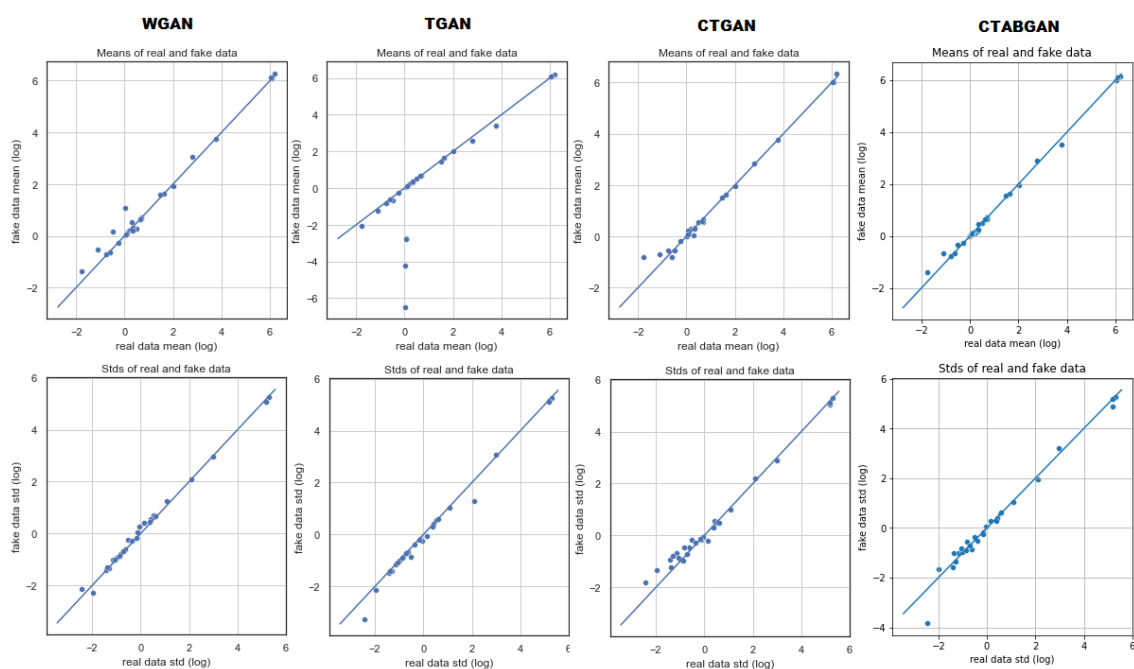


Figure A.1: The first row illustrates the log transformations of mean and the second row depicts the log transformations of standard deviation of each column

A.2 Pair-wise Correlation

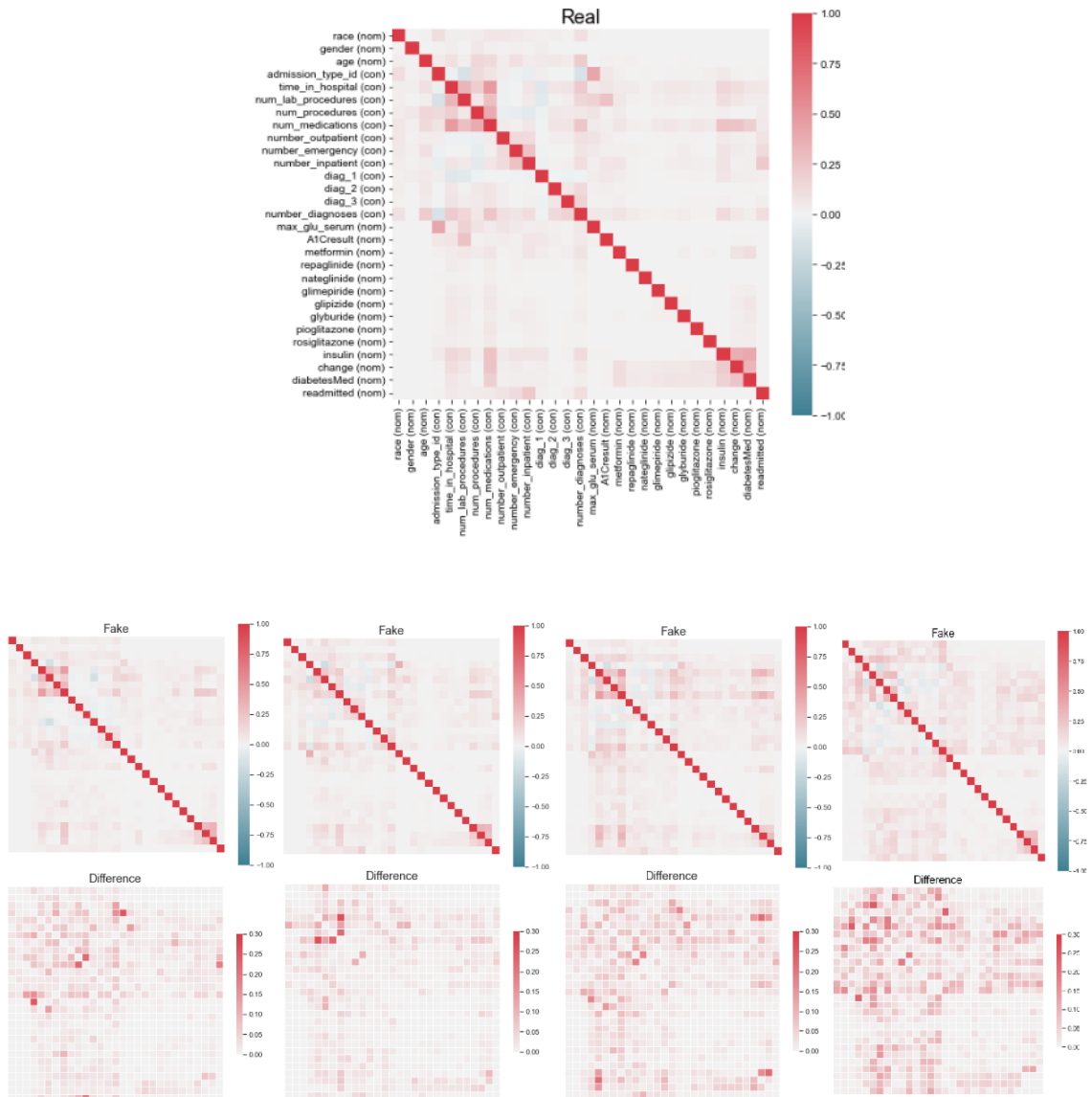


Figure A.2: The correlation matrices of real and synthetic datasets and their difference. Figures from left to right belongs to: WGAN, TGAN, CTGAN, CTABGAN

A.3 Column Distributions

A.3.1 Continuous Columns

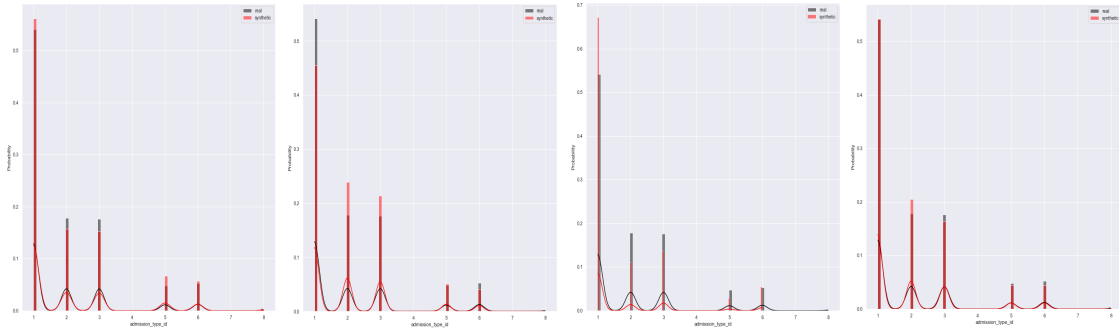


Figure A.3: The probability distributions of admission_type_id

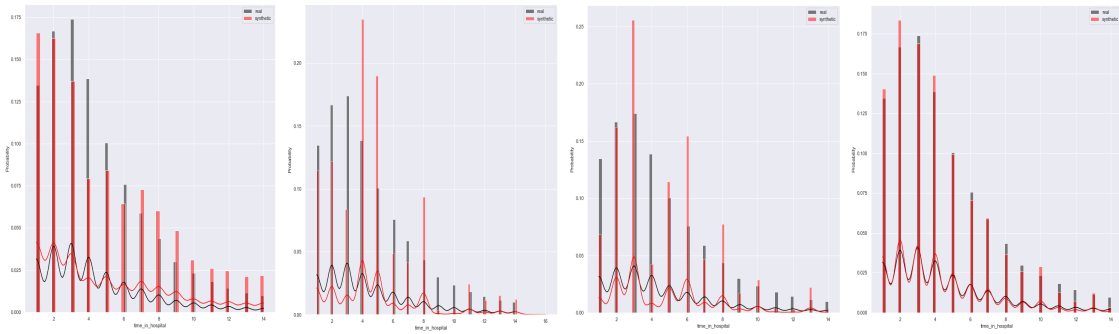


Figure A.4: The probability distributions of time_in_hospital

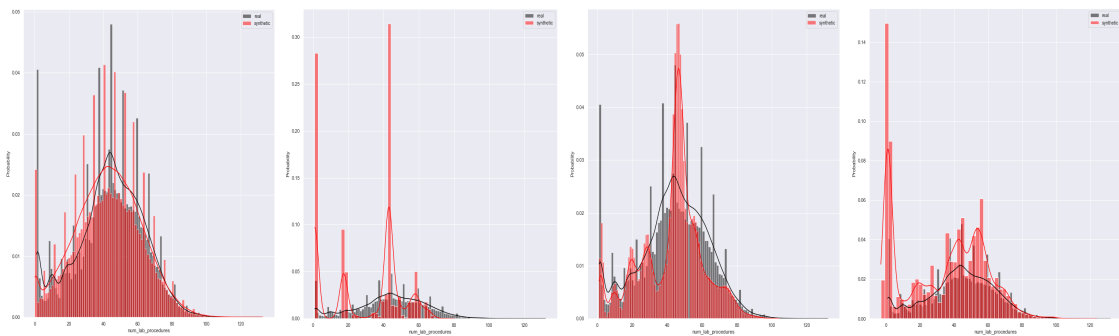


Figure A.5: The probability distributions of num_lab_procedures

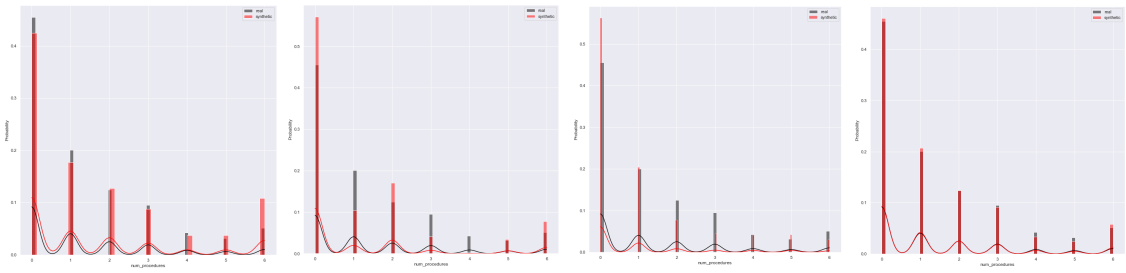


Figure A.6: The probability distributions of num_procedures

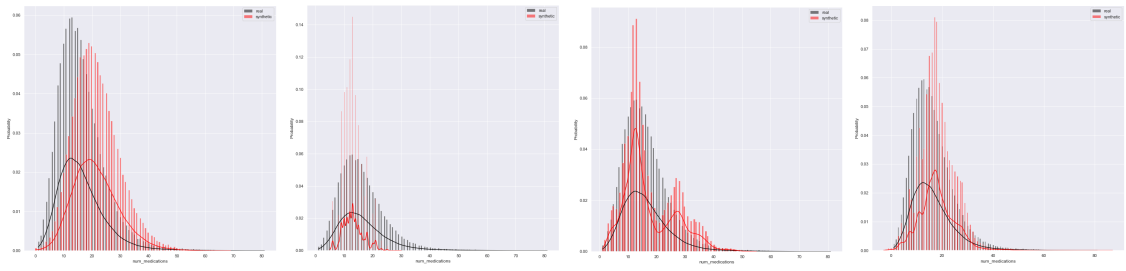


Figure A.7: The probability distributions of num_medications

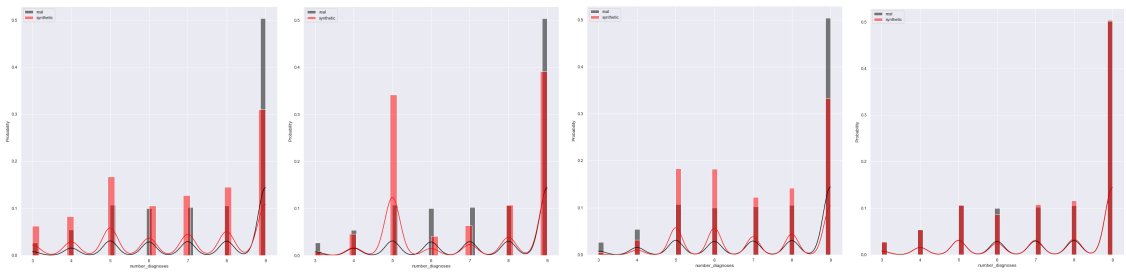


Figure A.8: The probability distributions of num_diagnoses

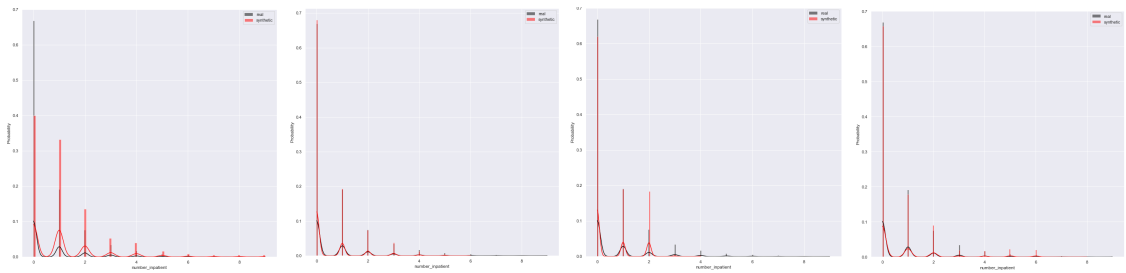


Figure A.9: The probability distributions of number_inpatient

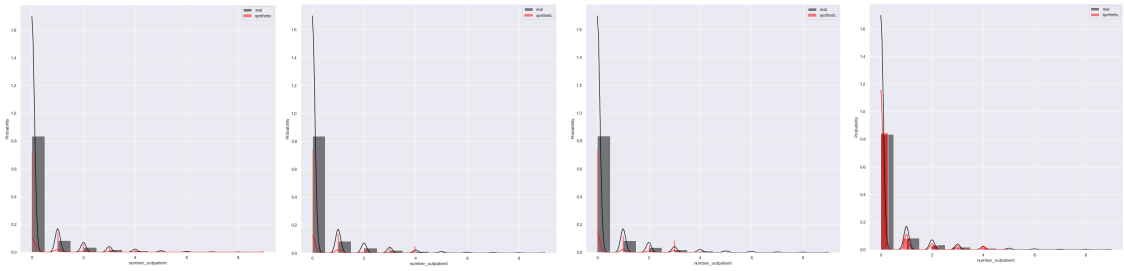


Figure A.10: The probability distributions of number_outpatient

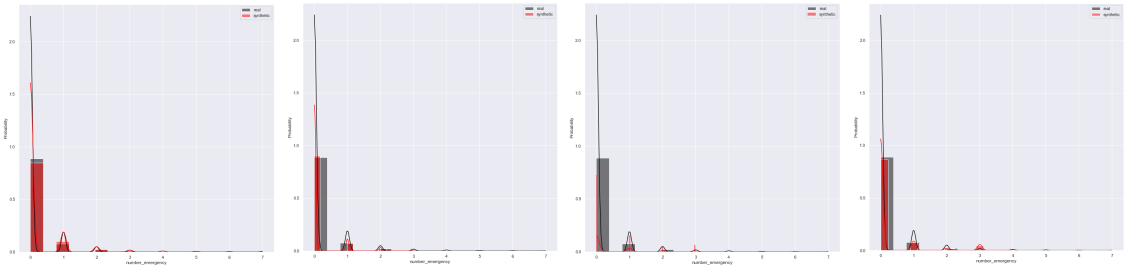


Figure A.11: The probability distributions of number_emergency

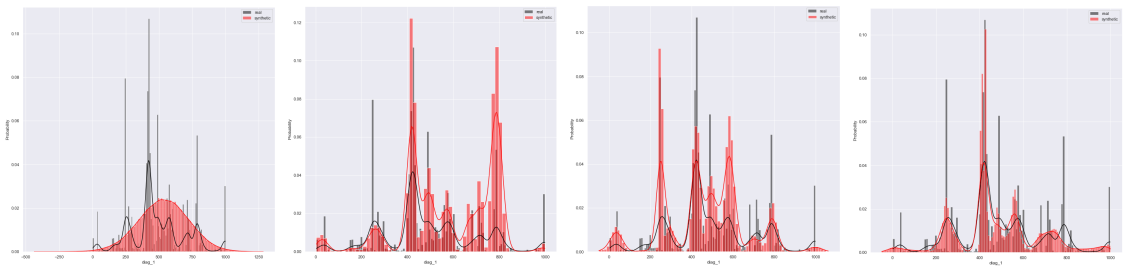


Figure A.12: The probability distributions of diag_1

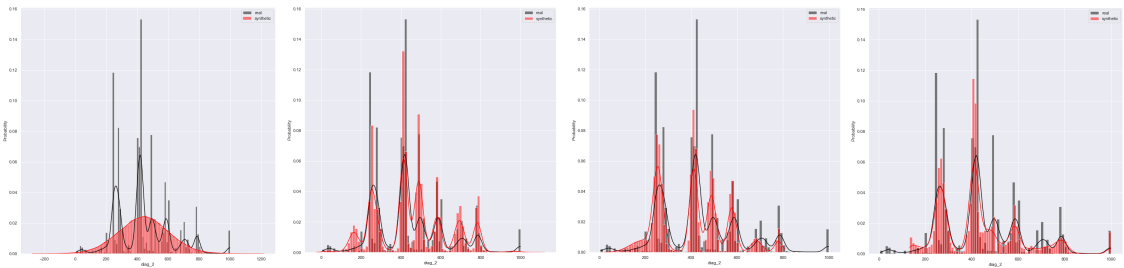


Figure A.13: The probability distributions of diag_2

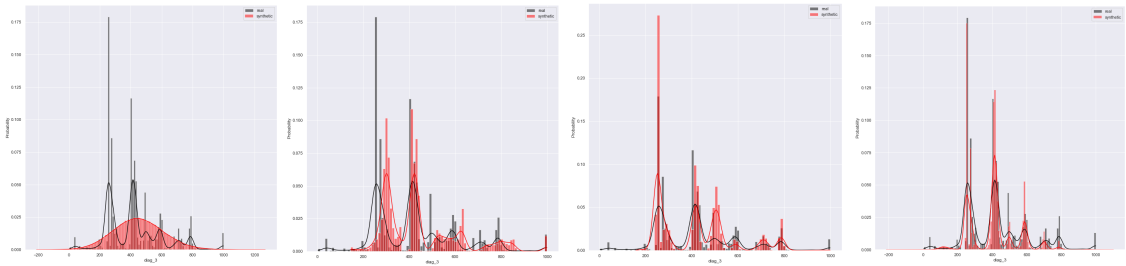


Figure A.14: The probability distributions of diag_3

A.3.2 Categorical Columns

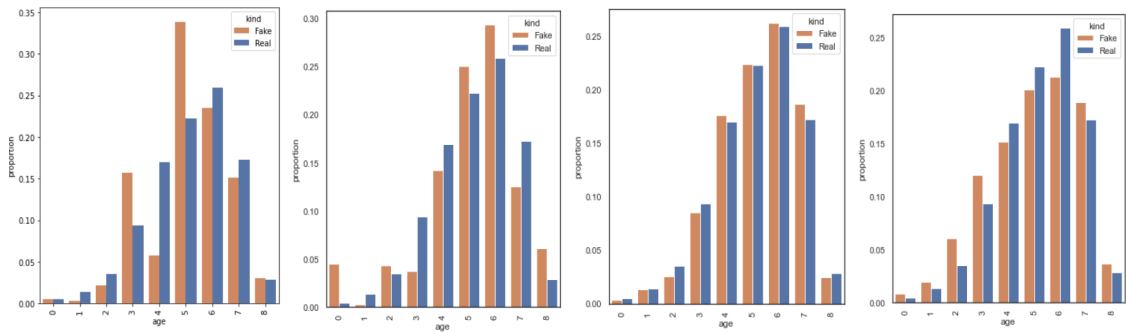


Figure A.15: The probability distributions of age

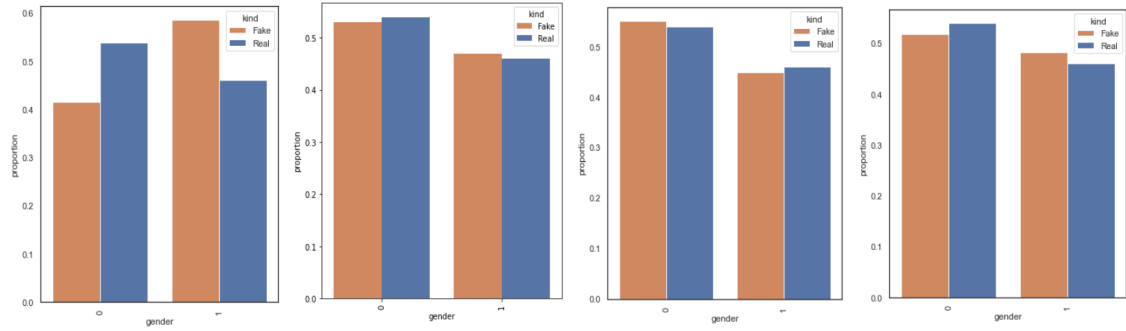


Figure A.16: The probability distributions of gender

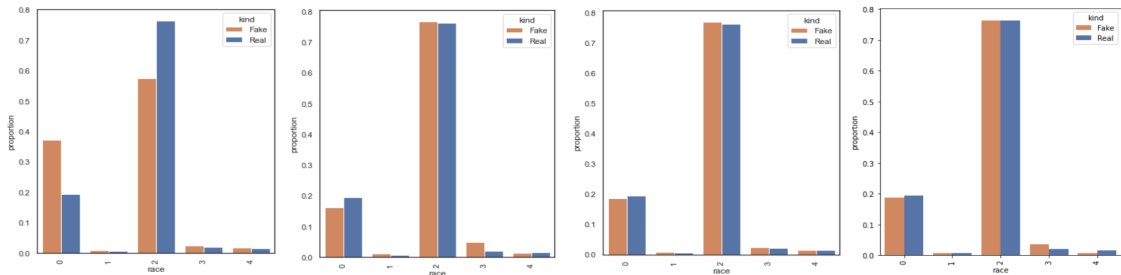


Figure A.17: The probability distributions of race

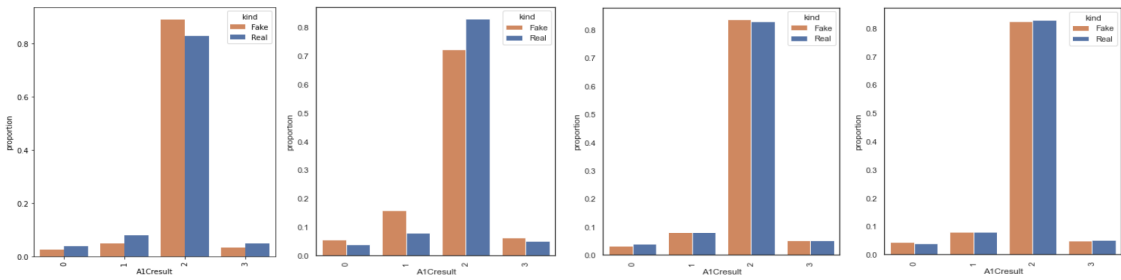


Figure A.18: The probability distributions of A1Cresult

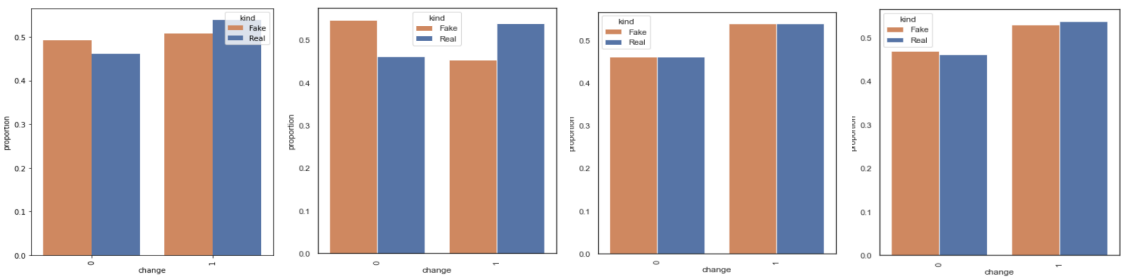


Figure A.19: The probability distributions of change

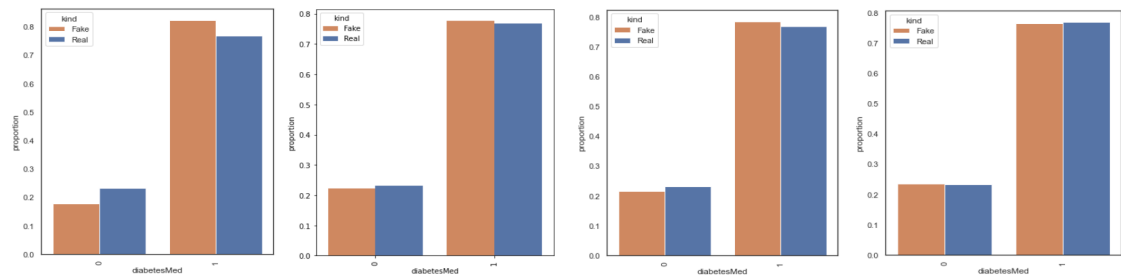


Figure A.20: The probability distributions of diabetesMed

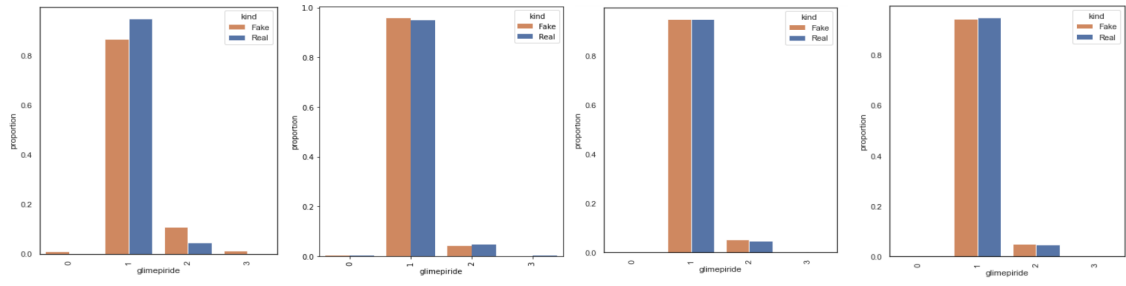


Figure A.21: The probability distributions of glimepiride

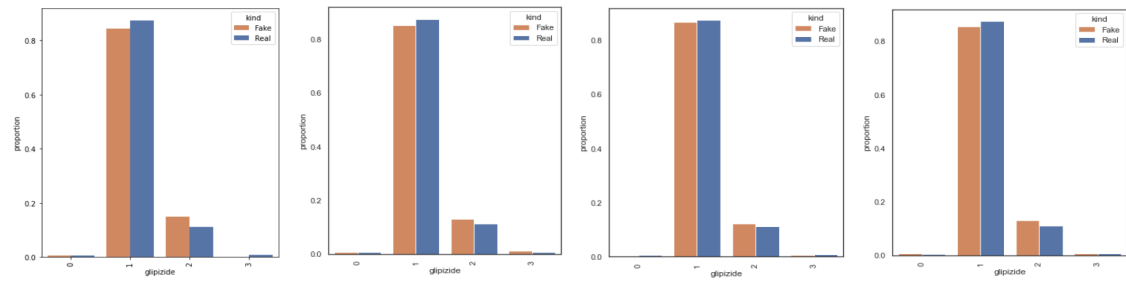


Figure A.22: The probability distributions of glipizide

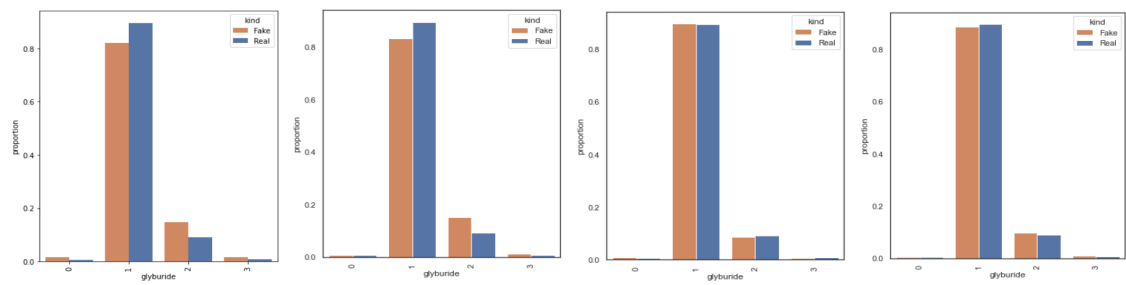


Figure A.23: The probability distributions of glyburide

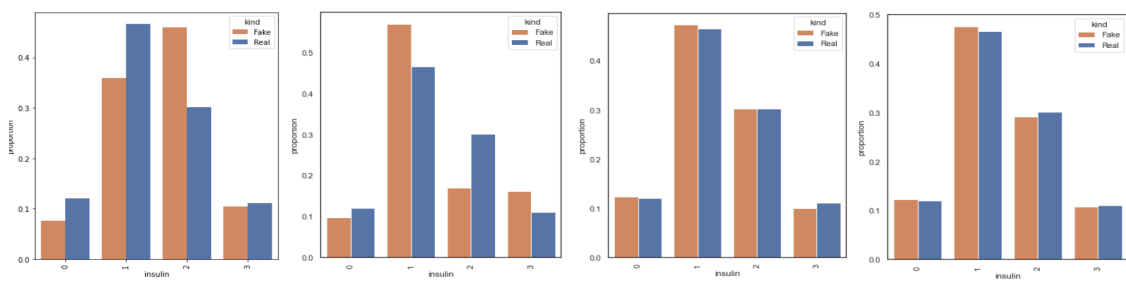


Figure A.24: The probability distributions of insulin

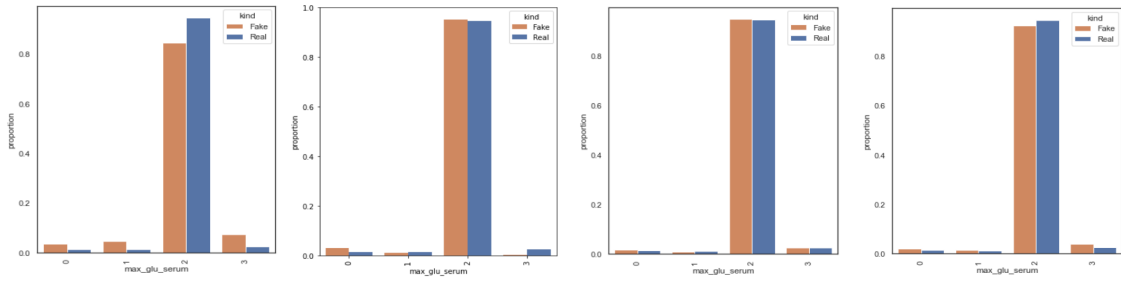


Figure A.25: The probability distributions of max_glu_serum

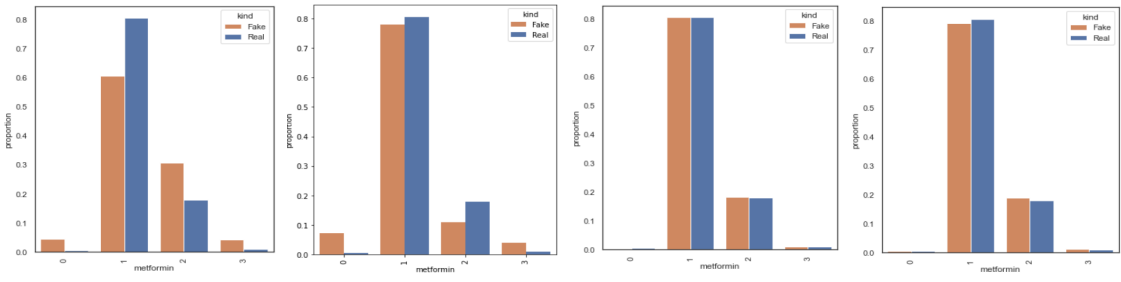


Figure A.26: The probability distributions of metformin

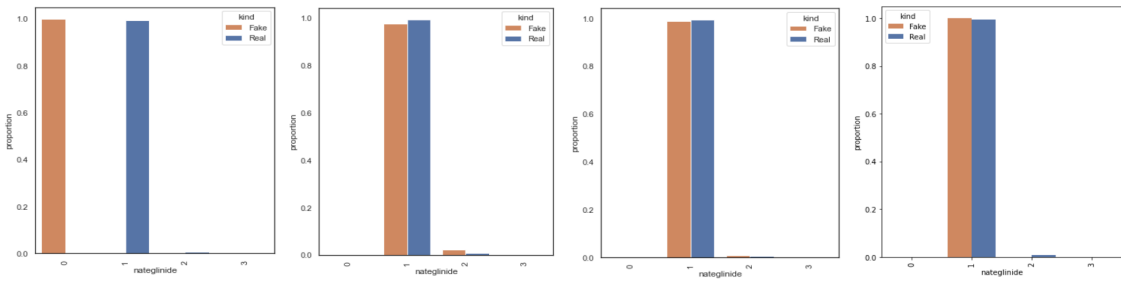


Figure A.27: The probability distributions of nateglinide

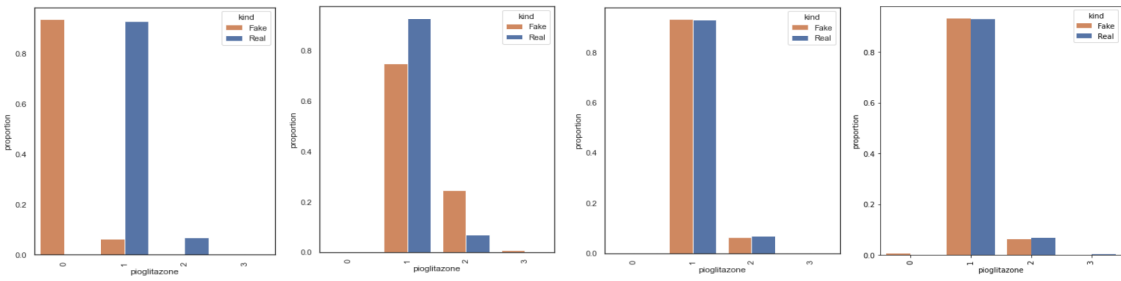


Figure A.28: The probability distributions of pioglitazone

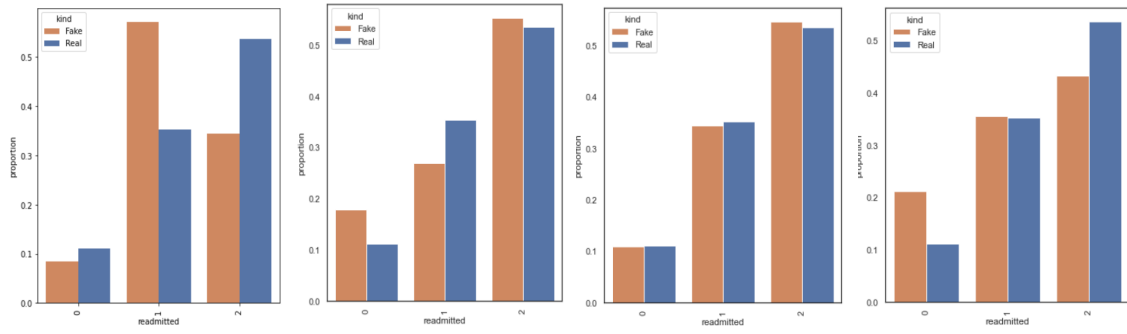


Figure A.29: The probability distributions of readmitted

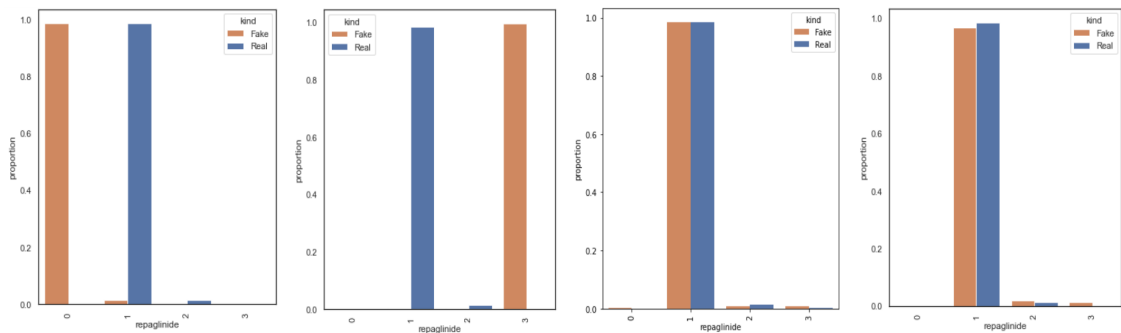


Figure A.30: The probability distributions of repaglinide

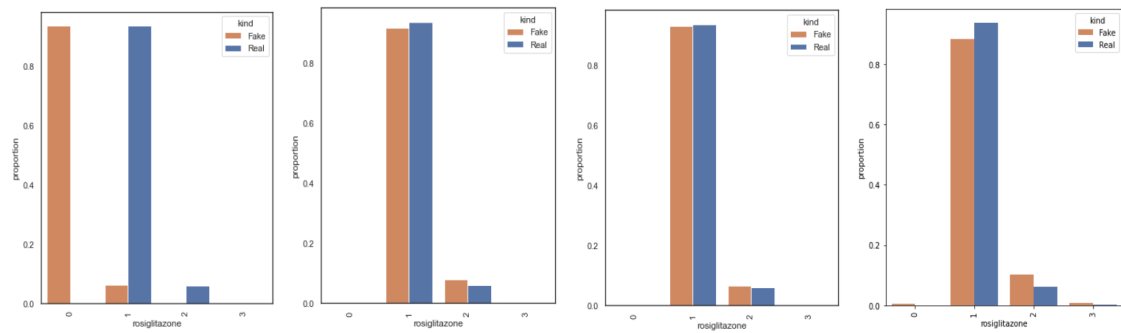


Figure A.31: The probability distributions of rosiglitazone

A.4 General Utility Scores

Table A.1: General utility scores. Columns from left to right represent: basic similarity score, Kolmogorov–Smirnov test score, Chi-squared test score, mean and standard deviation of the DCR distribution between synthetic and original records, correlation distance, Logistic Regression and SVM classification scores.

Model	Basic	KS test	CS test	DCR(μ)	DCR(σ)	Corr	LR	SVC
WGAN	0.960	0.75	0.78	3.10	0.46	1.00	0.33	0.11
TGAN	0.948	0.87	0.95	2.69	0.61	0.87	0.52	0.29
CTGAN	0.987	0.87	0.99	2.71	0.64	1.29	0.54	0.33
CTABGAN	0.976	0.90	0.97	3.02	0.46	1.65	0.70	0.56

A.5 ML Cross-Testing Scores

Table A.2: F1-scores of each classifier trained on the original and synthetic training datasets and evaluated on the real test data. DT, RF, LR and MLP stand for Decision Tree, Random Forest, Logistic Regression and Multi-layer Perceptron.

Classifiers	DT		RF		LR		MLP	
	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>
WGAN	0.456	0.373	0.522	0.400	0.569	0.440	0.573	0.430
TGAN	0.456	0.400	0.522	0.440	0.569	0.519	0.573	0.486
CTGAN	0.456	0.426	0.522	0.478	0.569	0.523	0.573	0.529
CTABGAN	0.456	0.440	0.522	0.499	0.569	0.560	0.573	0.560

Table A.3: The difference of the F1-scores of each classifier trained on the original and synthetic datasets.

Model	Δ F1-DT	Δ F1-RF	Δ F1-LR	Δ F1-MLP
WGAN	0.083	0.122	0.129	0.143
TGAN	0.056	0.082	0.050	0.087
CTGAN	0.030	0.044	0.046	0.044
CTABGAN	0.016	0.023	0.009	0.013

Appendix B

MIMIC III

B.1 Basic Similarities

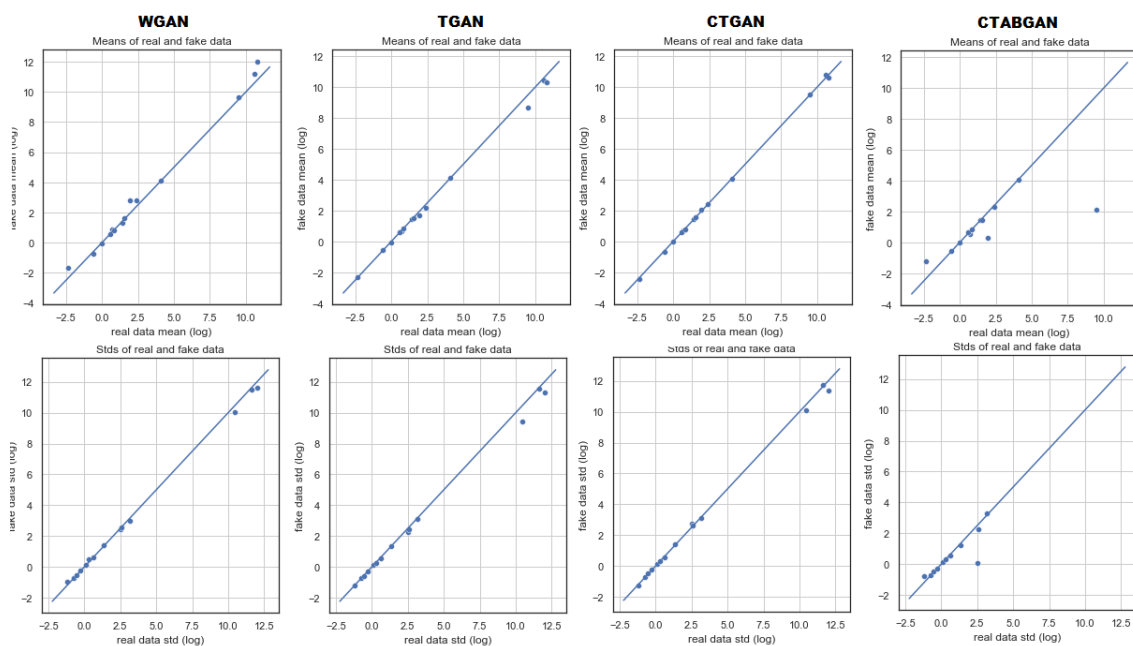


Figure B.1: The first row illustrates the log transformations of mean and the second row depicts the log transformations of standard deviation of each column

B.2 Pair-wise Correlation

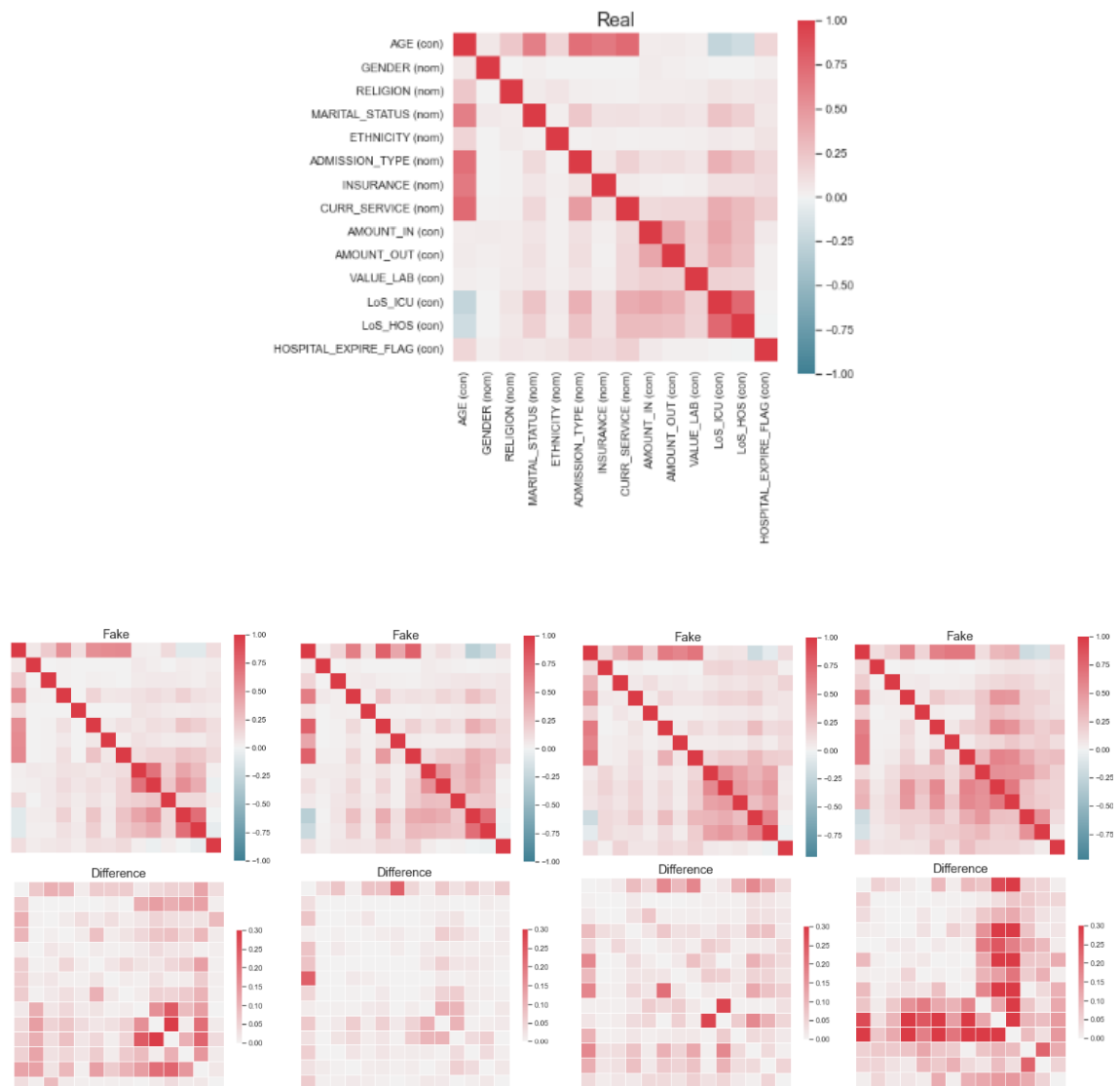


Figure B.2: The correlation matrices of real and synthetic datasets and their difference. Figures from left to right belongs to: WGAN, TGAN, CTGAN, CTABGAN

B.3 Column Distributions

B.3.1 Continuous Columns

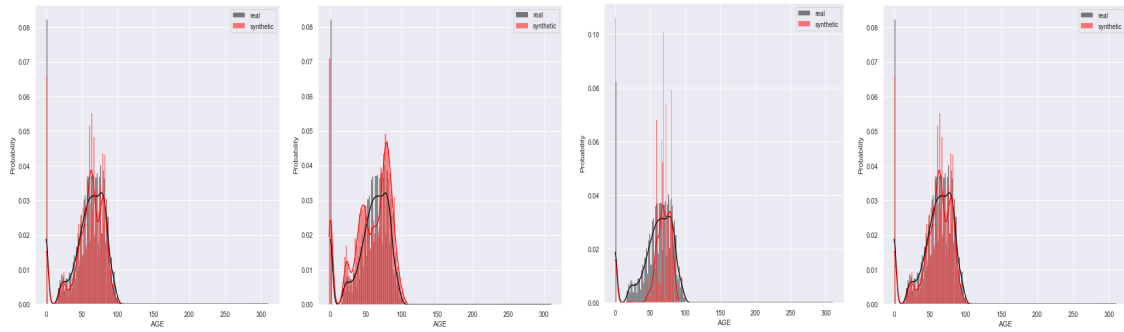


Figure B.3: The probability distributions of age

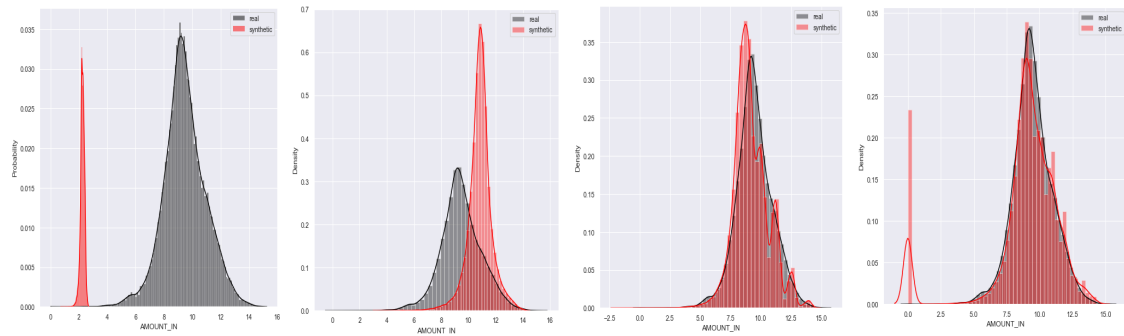


Figure B.4: The probability distributions of amount_in

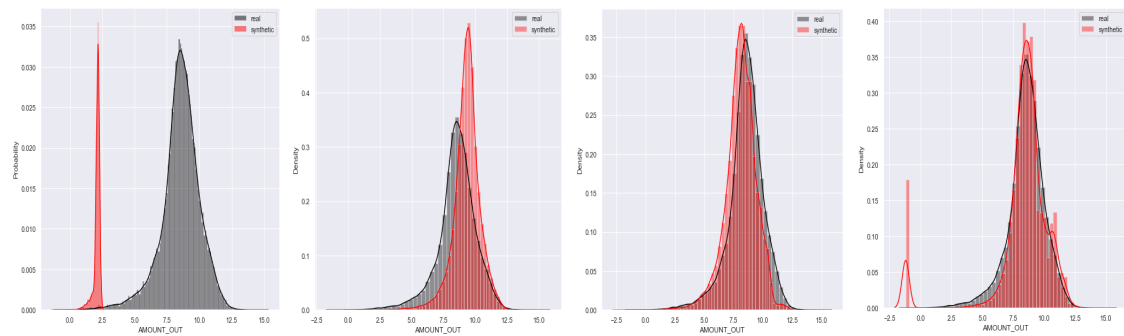


Figure B.5: The probability distributions of amount_out

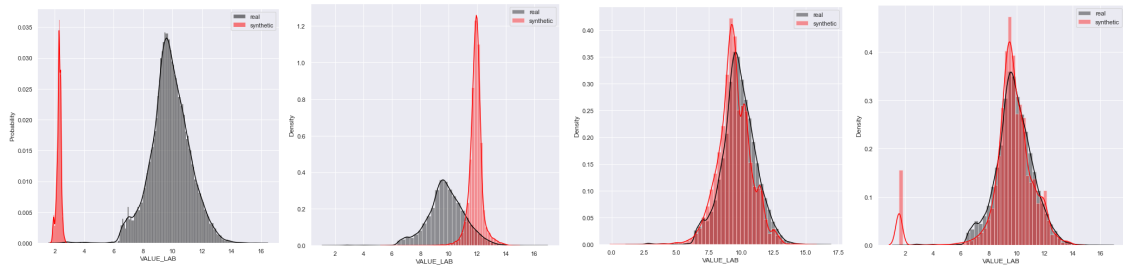


Figure B.6: The probability distributions of value_lab

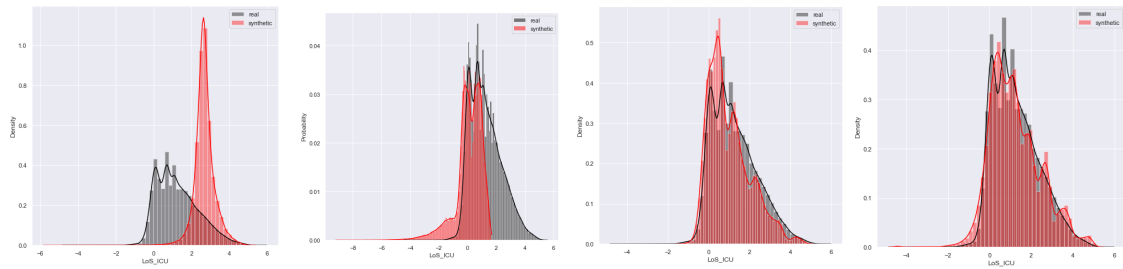


Figure B.7: The probability distributions of los_icu

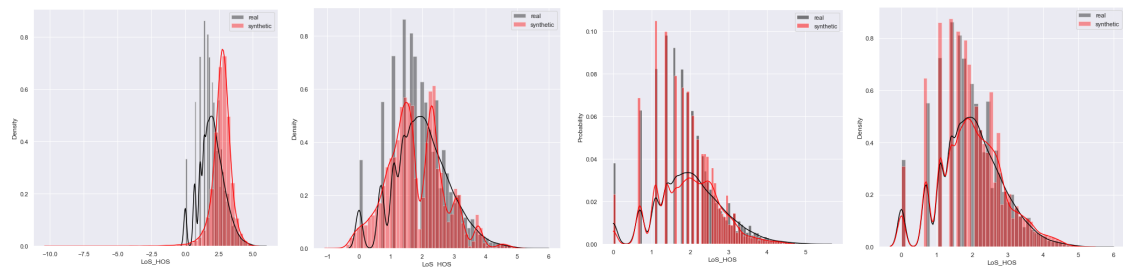


Figure B.8: The probability distributions of los_hos

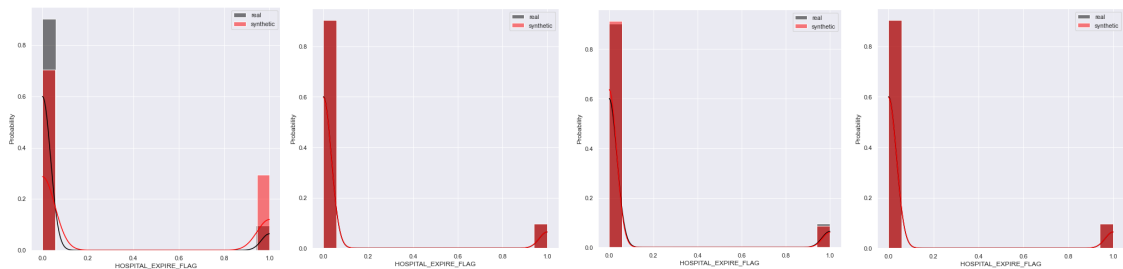


Figure B.9: The probability distributions of hospital_expire_flag

B.3.2 Categorical Columns

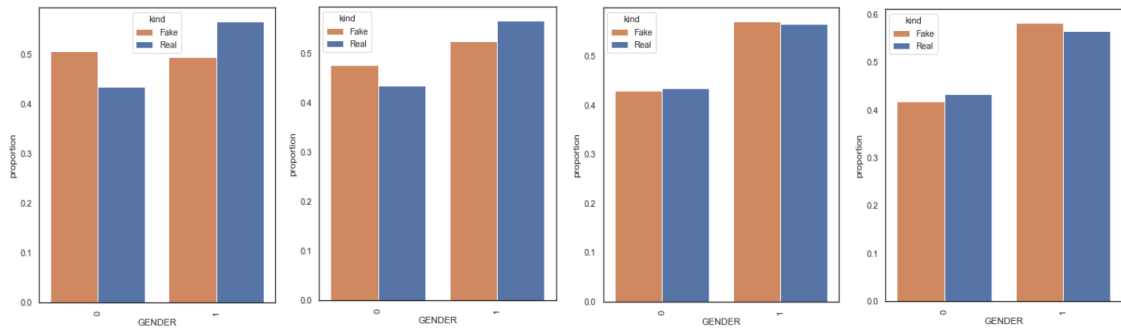


Figure B.10: The probability distributions of gender

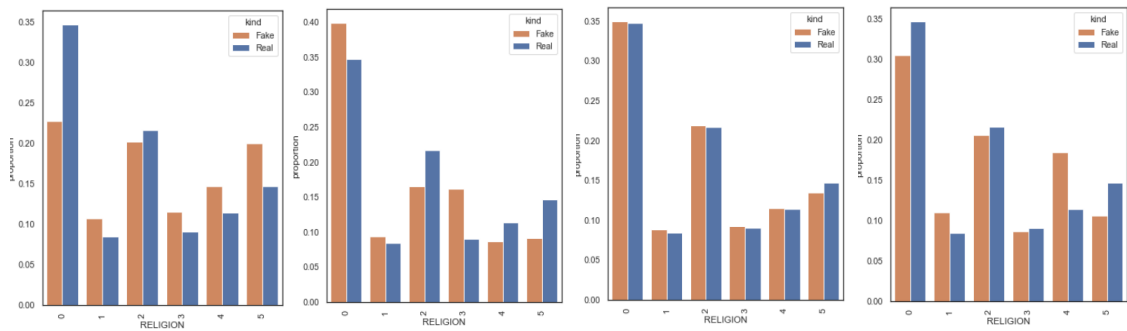


Figure B.11: The probability distributions of religion

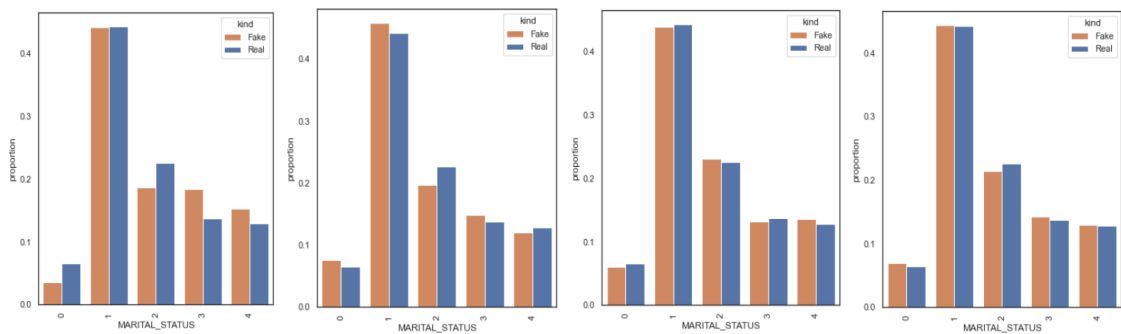


Figure B.12: The probability distributions of marital_status

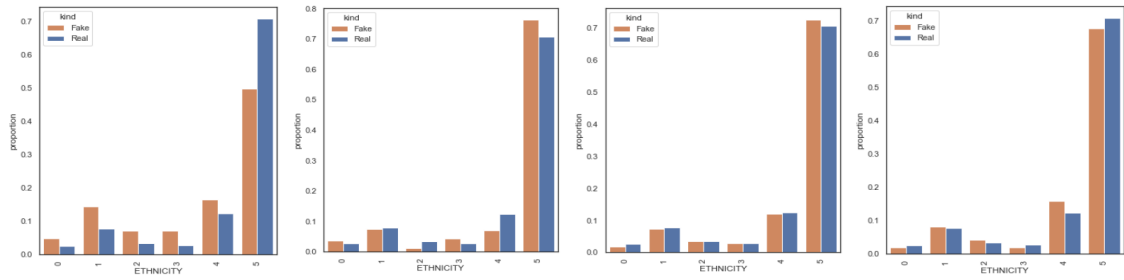


Figure B.13: The probability distributions of ethnicity

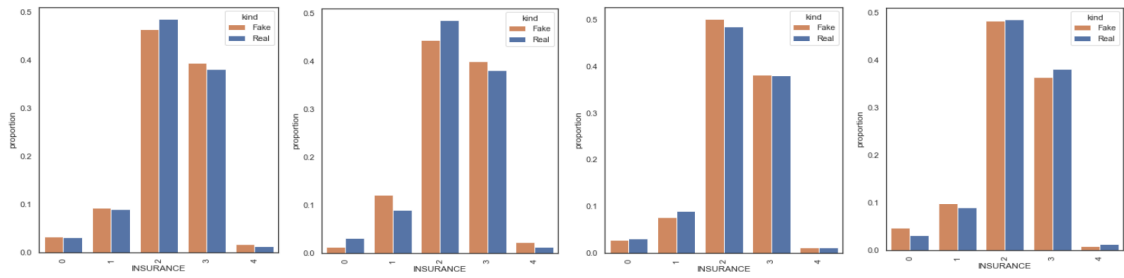


Figure B.14: The probability distributions of insurance

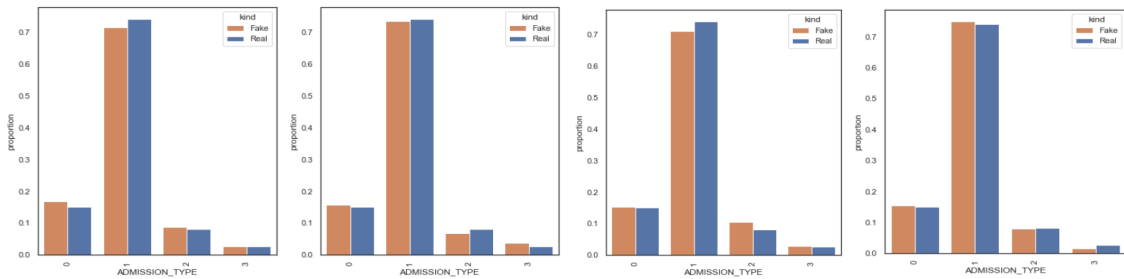


Figure B.15: The probability distributions of admission_type

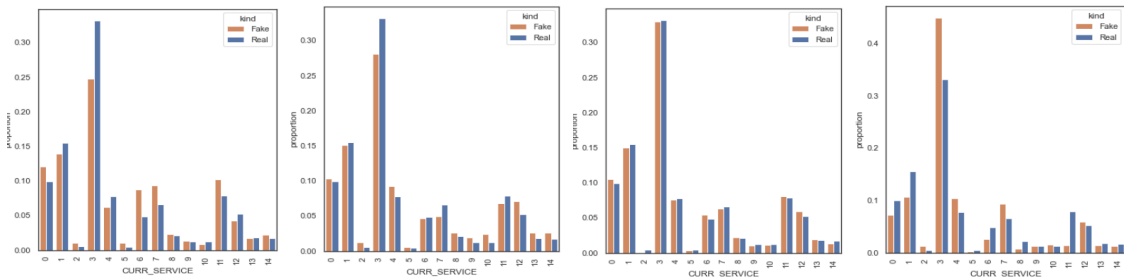


Figure B.16: The probability distributions of curr service

B.4 General Utility Scores

Table B.1: General utility scores. Columns from left to right represent: basic similarity score, Kolmogorov–Smirnov test score, Chi-squared test score, mean and standard deviation of the DCR distribution between synthetic and original records, correlation distance, Logistic Regression and SVM classification scores.

Model	Basic	KS test	CS test	DCR(μ)	DCR(σ)	Corr	LR	SVC
WGAN	0.975	0.468	0.94	1.11	0.84	1.09	0.22	0.11
TGAN	0.967	0.729	0.97	0.93	0.80	0.65	0.62	0.39
CTGAN	0.978	0.844	0.99	0.97	0.90	0.95	0.73	0.48
CTABGAN	0.951	0.960	0.97	1.37	1.14	2.13	0.79	0.54

B.5 ML Cross-Testing Scores

Table B.2: F1-scores of each classifier trained on the original and synthetic training datasets and evaluated on the real test data. DT, RF, LR and MLP stand for Decision Tree, Random Forest, Logistic Regression and Multi-layer Perceptron.

Classifiers	DT		RF		LR		MLP	
	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>
WGAN	0.866	0.677	0.910	0.790	0.900	0.806	0.732	0.717
TGAN	0.866	0.798	0.910	0.880	0.900	0.901	0.732	0.767
CTGAN	0.866	0.814	0.910	0.890	0.900	0.915	0.732	0.693
CTABGAN	0.866	0.835	0.910	0.920	0.900	0.893	0.732	0.704

Table B.3: Difference of F1-scores of each classifier trained on the original and synthetic datasets.

Model	Δ F1-DT	Δ F1-RF	Δ F1-LR	Δ F1-MLP
WGAN	0.189	0.120	0.094	0.015
TGAN	0.068	0.030	0.001	0.035
CTGAN	0.052	0.020	0.015	0.039
CTABGAN	0.031	0.010	0.007	0.028

Appendix C

Thyroid

C.1 Basic Similarities

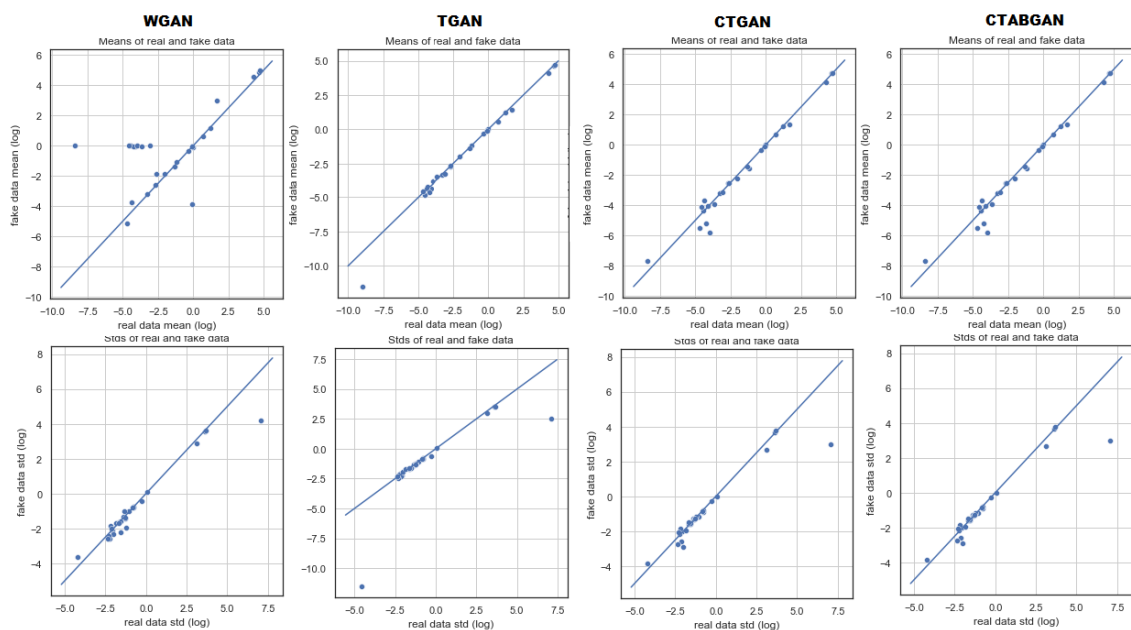


Figure C.1: The first row illustrates the log transformations of mean and the second row depicts the log transformations of standard deviation of each column

C.2 Pair-wise Correlation

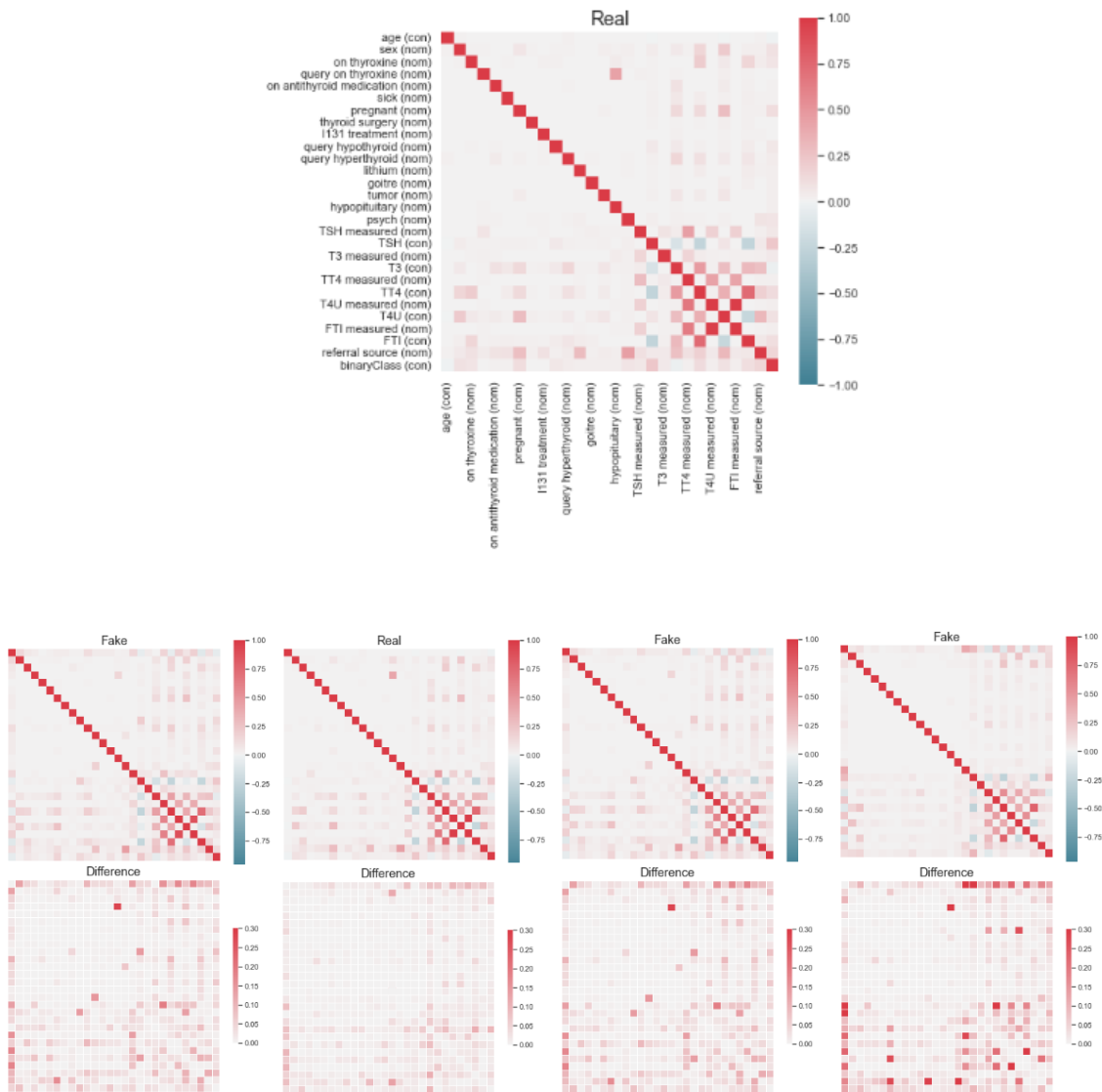


Figure C.2: The correlation matrices of real and synthetic datasets and their difference. Figures from left to right belongs to: WGAN, TGAN, CTGAN, CTABGAN

C.3 Column Distributions

C.3.1 Continuous Columns

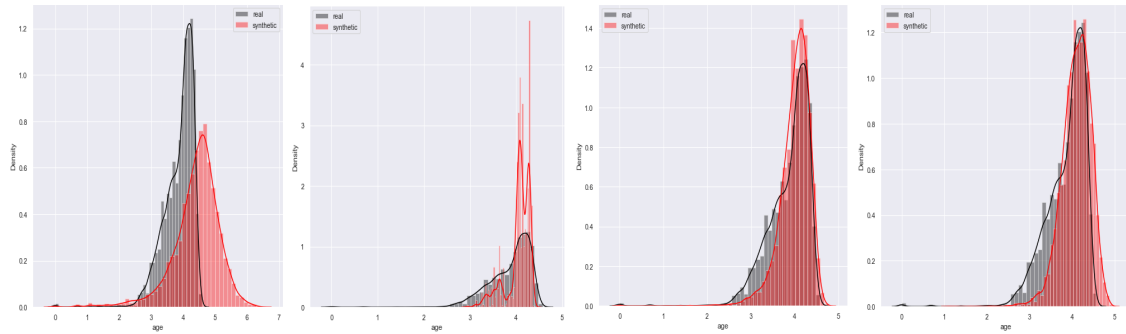


Figure C.3: The probability distributions of age

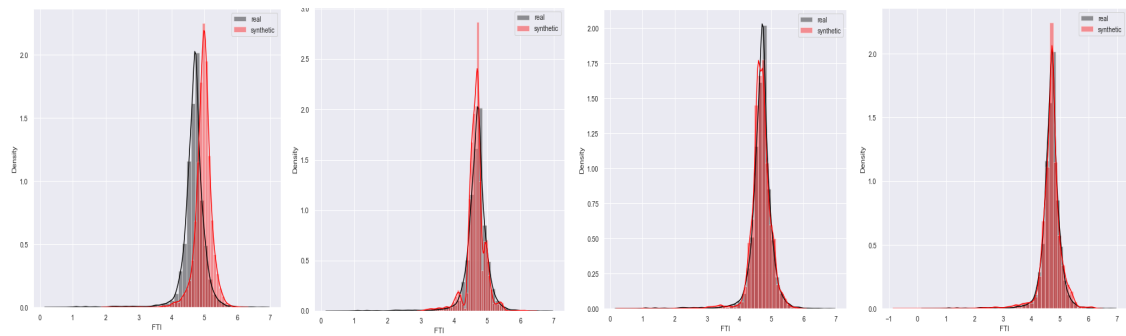


Figure C.4: The probability distributions of FTI

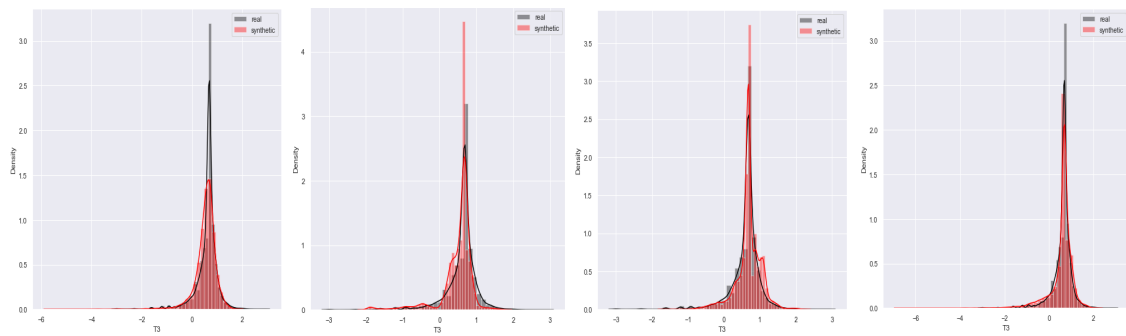


Figure C.5: The probability distributions of T3

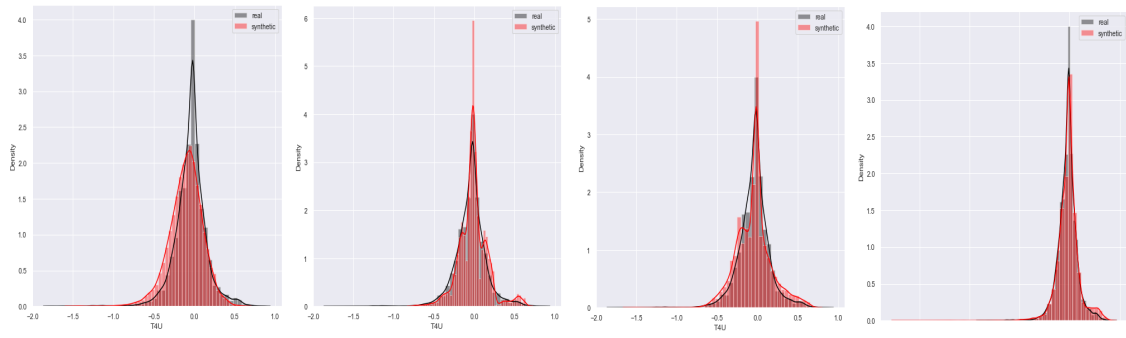


Figure C.6: The probability distributions of T4U

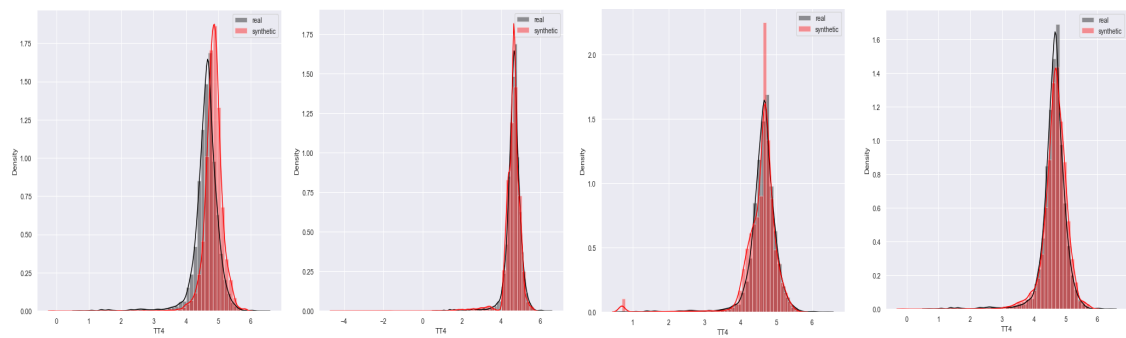


Figure C.7: The probability distributions of TT4

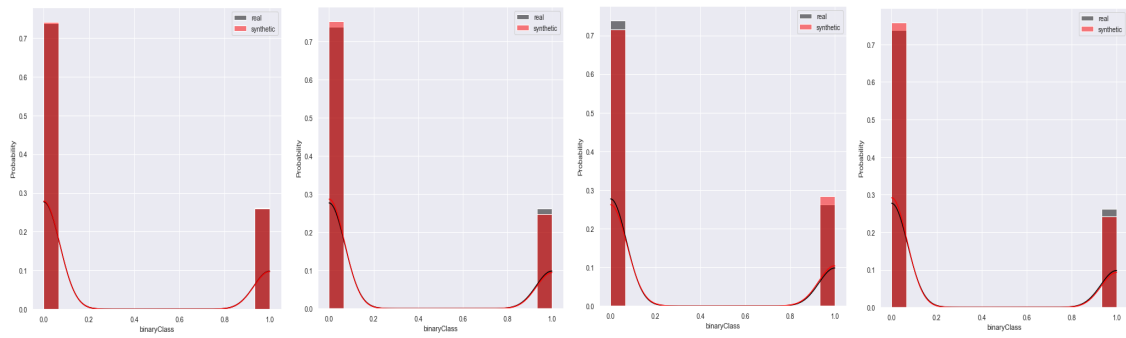


Figure C.8: The probability distributions of binaryClass

C.3.2 Categorical Columns

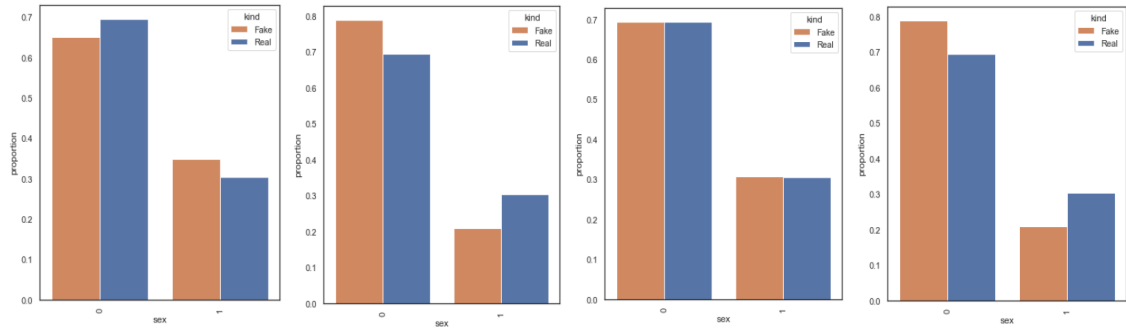


Figure C.9: The probability distributions of sex

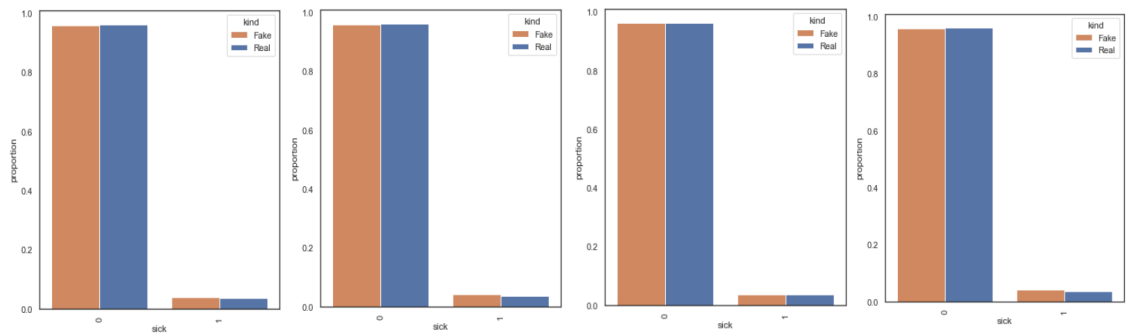


Figure C.10: The probability distributions of sick

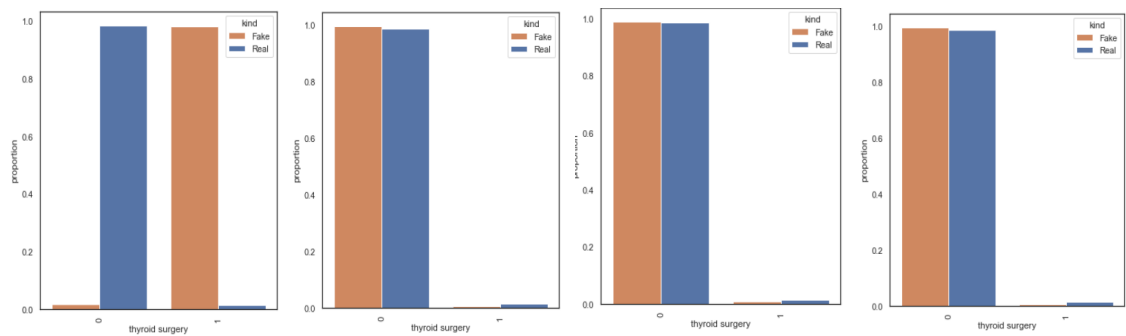


Figure C.11: The probability distributions of thyroid surgery

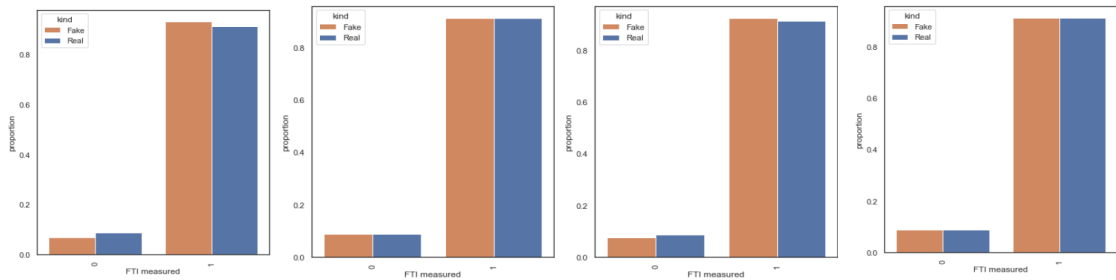


Figure C.12: The probability distributions of FTI measured

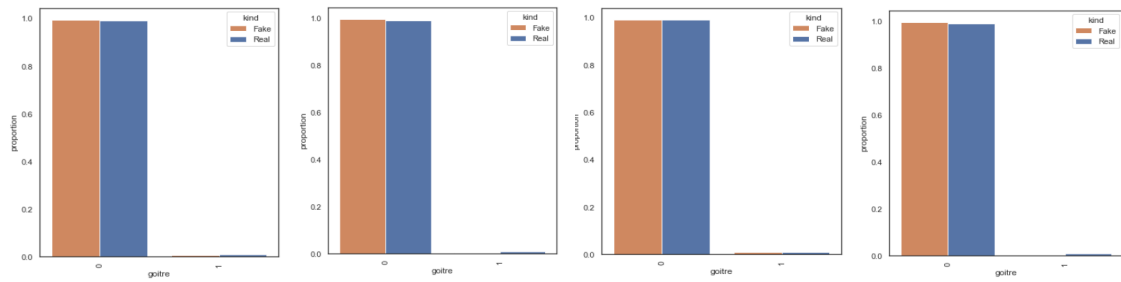


Figure C.13: The probability distributions of goitre

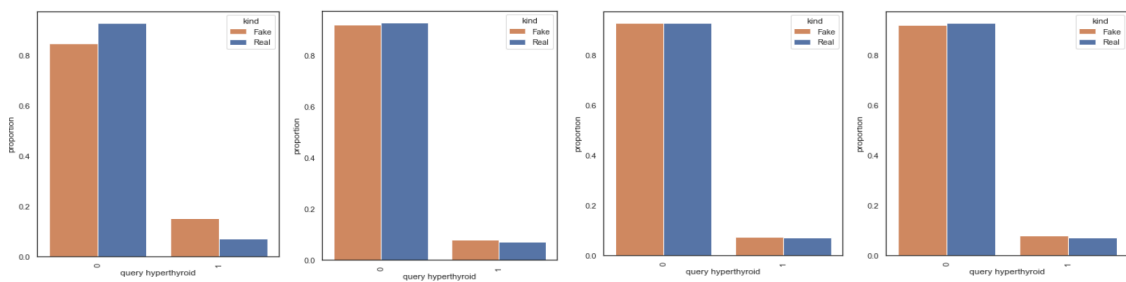


Figure C.14: The probability distributions of query hyperthyroid

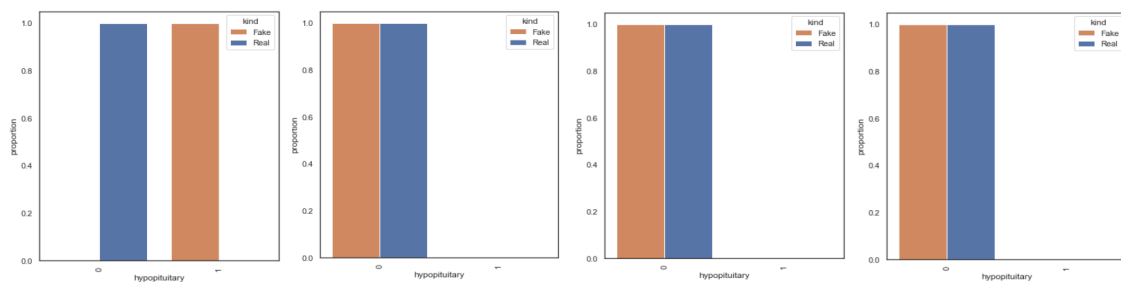


Figure C.15: The probability distributions of hypopituitary

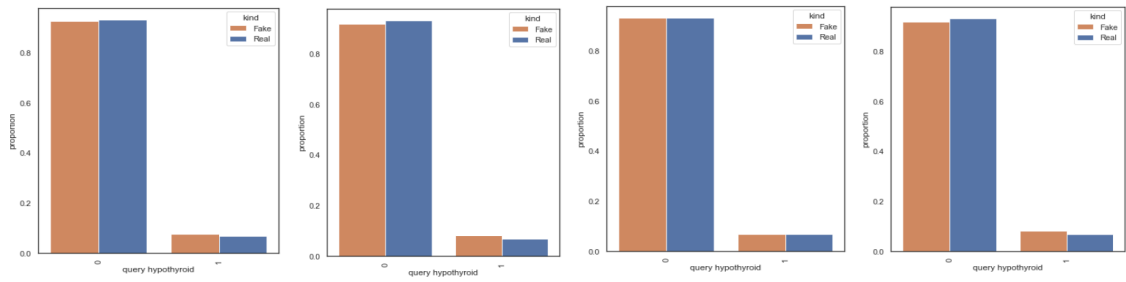


Figure C.16: The probability distributions of query hypothyroid

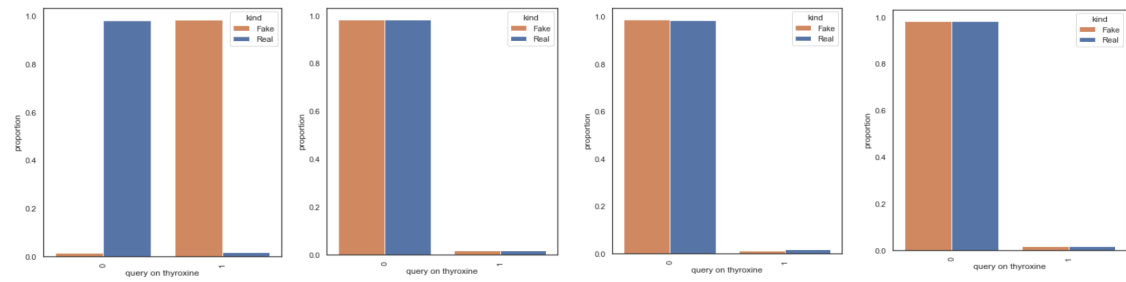


Figure C.17: The probability distributions of query on thyroxine

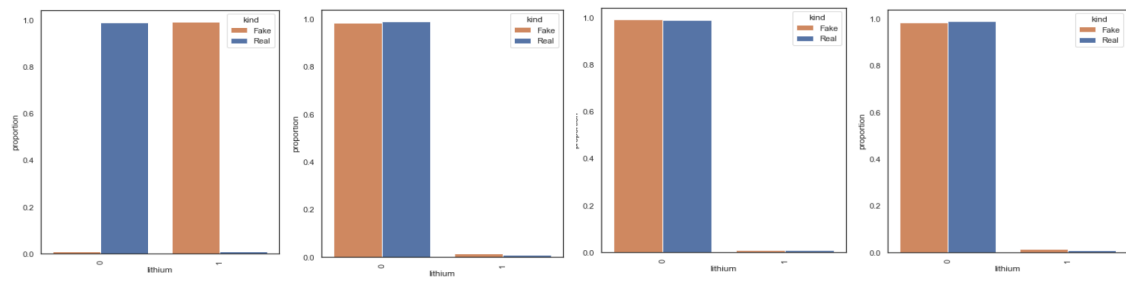


Figure C.18: The probability distributions of lithium

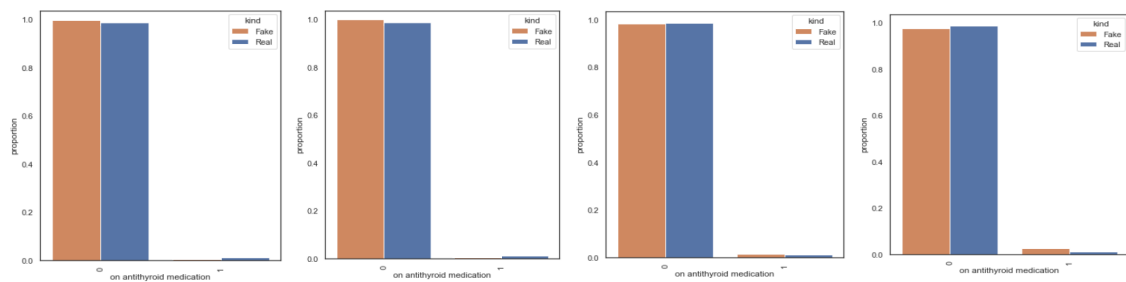


Figure C.19: The probability distributions of on antithyroid medication

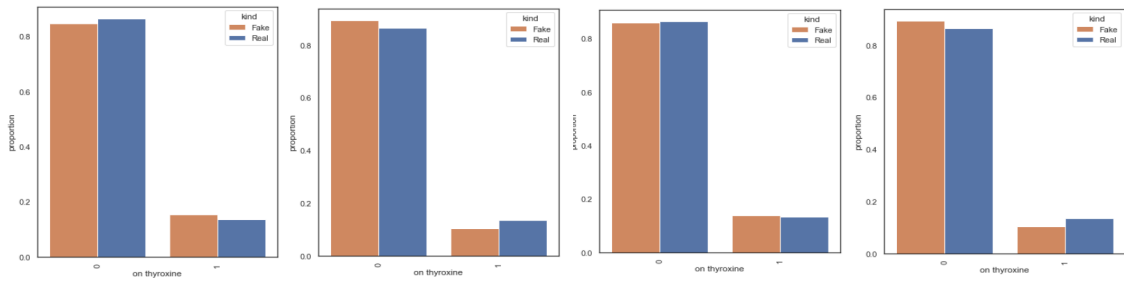


Figure C.20: The probability distributions of on thyroxine

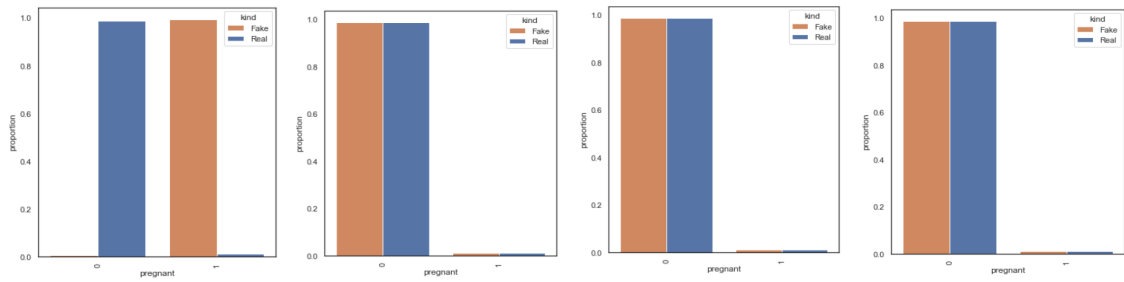


Figure C.21: The probability distributions of pregnant

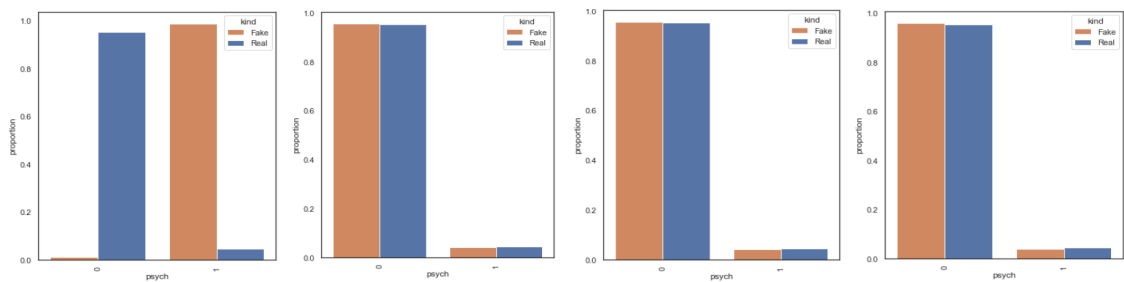


Figure C.22: The probability distributions of psych

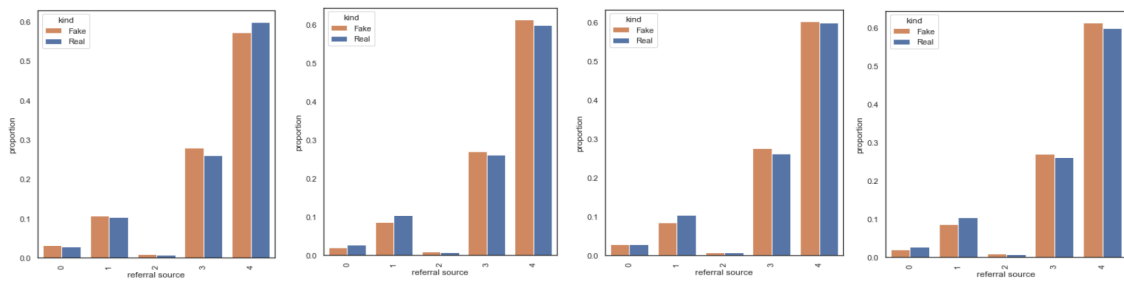


Figure C.23: The probability distributions of referral source

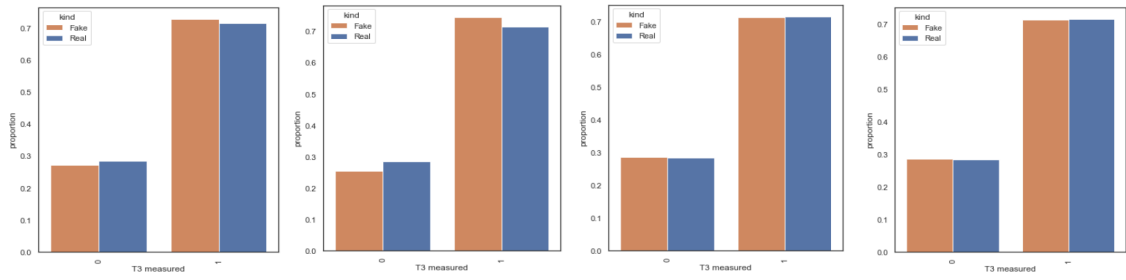


Figure C.24: The probability distributions of T3 measured

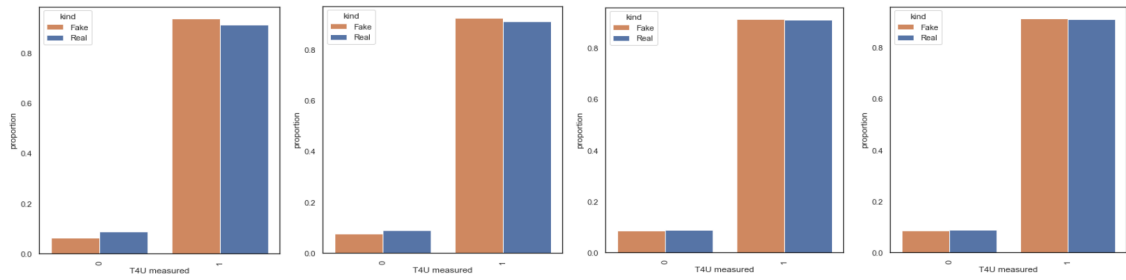


Figure C.25: The probability distributions of T4U measured

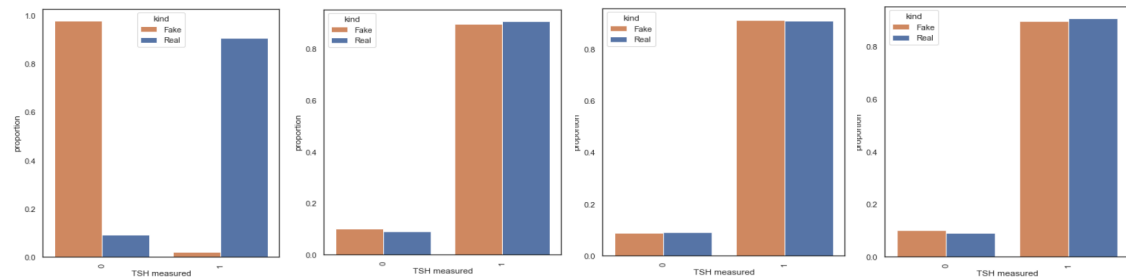


Figure C.26: The probability distributions of TSH measured

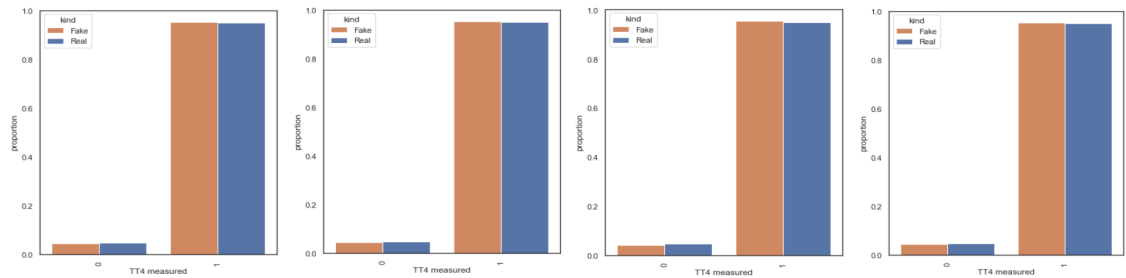


Figure C.27: The probability distributions of TT4 measured

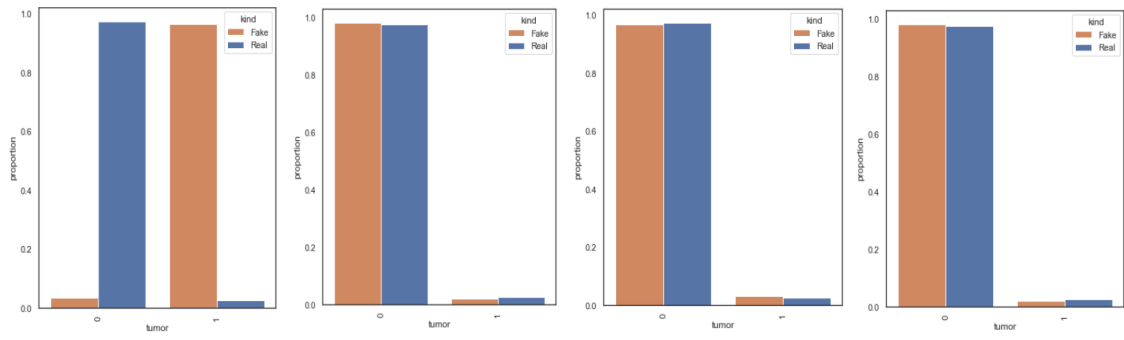


Figure C.28: The probability distributions of tumor

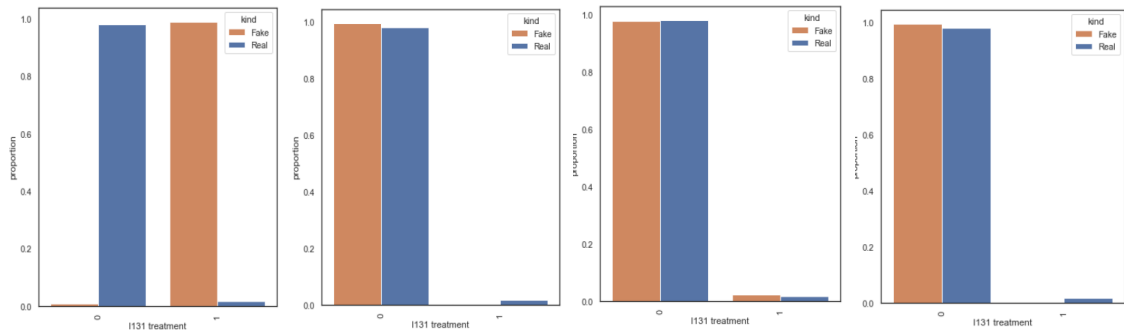


Figure C.29: The probability distributions of treatment

C.4 General Utility Scores

Table C.1: General utility scores. Columns from left to right represent: basic similarity score, Kolmogorov–Smirnov test score, Chi-squared test score, mean and standard deviation of the DCR distribution between synthetic and original records, correlation distance, Logistic Regression and SVM classification scores.

Model	Basic	KS test	CS test	DCR(μ)	DCR(σ)	Corr	LR	SVC
WGAN	0.962	0.61	0.540	1.88	0.97	1.10	0.53	0.38
TGAN	0.981	0.82	0.879	1.39	1.05	0.55	0.70	0.60
CTGAN	0.970	0.89	0.980	1.34	1.08	1.65	0.70	0.53
CTABGAN	0.971	0.91	0.963	1.76	1.06	2.09	0.78	0.62

C.5 ML Cross-Testing Scores

Table C.2: F1-scores of each classifier trained on the original and synthetic training datasets and evaluated on the real test data. DT, RF, LR and MLP stand for Decision Tree, Random Forest, Logistic Regression and Multi-layer Perceptron.

Classifiers	DT		RF		LR		MLP	
	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>
WGAN	0.94	0.64	0.94	0.68	0.84	0.70	0.89	0.65
TGAN	0.94	0.70	0.94	0.72	0.84	0.75	0.89	0.72
CTGAN	0.94	0.74	0.94	0.75	0.84	0.79	0.89	0.77
CTABGAN	0.94	0.84	0.94	0.84	0.84	0.79	0.89	0.81

Table C.3: Difference of F1-scores of each classifier trained on the original and synthetic datasets.

Model	Δ F1-DT	Δ F1-RF	Δ F1-LR	Δ F1-MLP
WGAN	0.30	0.26	0.14	0.24
TGAN	0.24	0.22	0.09	0.17
CTGAN	0.20	0.19	0.05	0.12
CTABGAN	0.10	0.10	0.05	0.08

Appendix D

Epileptic

D.1 Basic Similarities

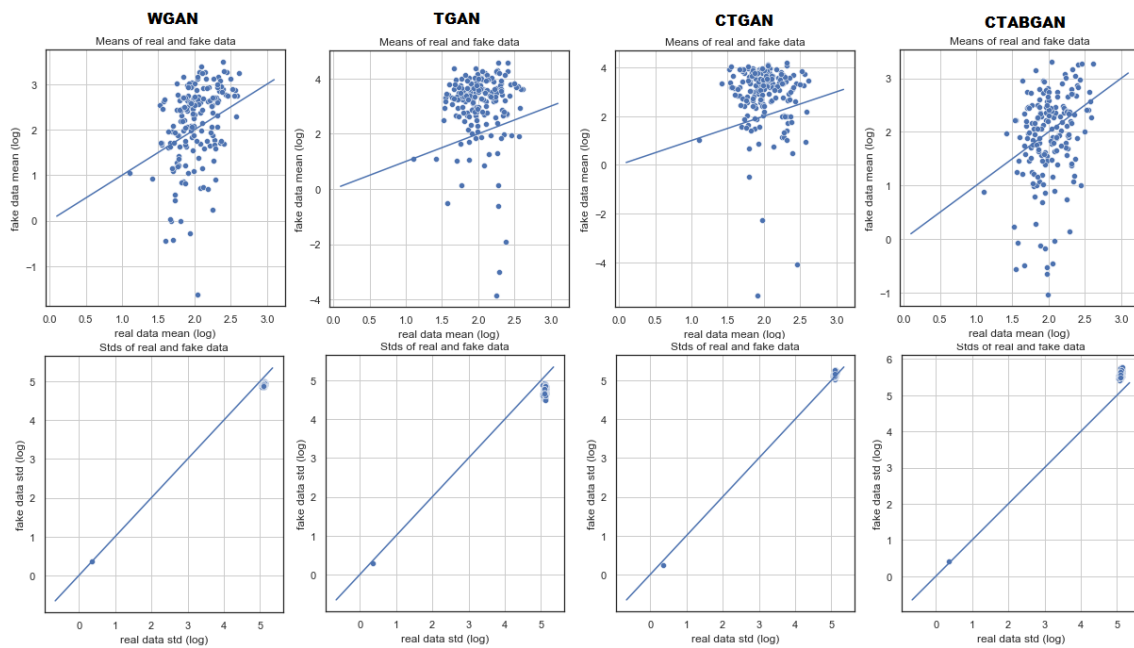


Figure D.1: The first row illustrates the log transformations of mean and the second row depicts the log transformations of standard deviation of each column

D.2 Pair-wise Correlation

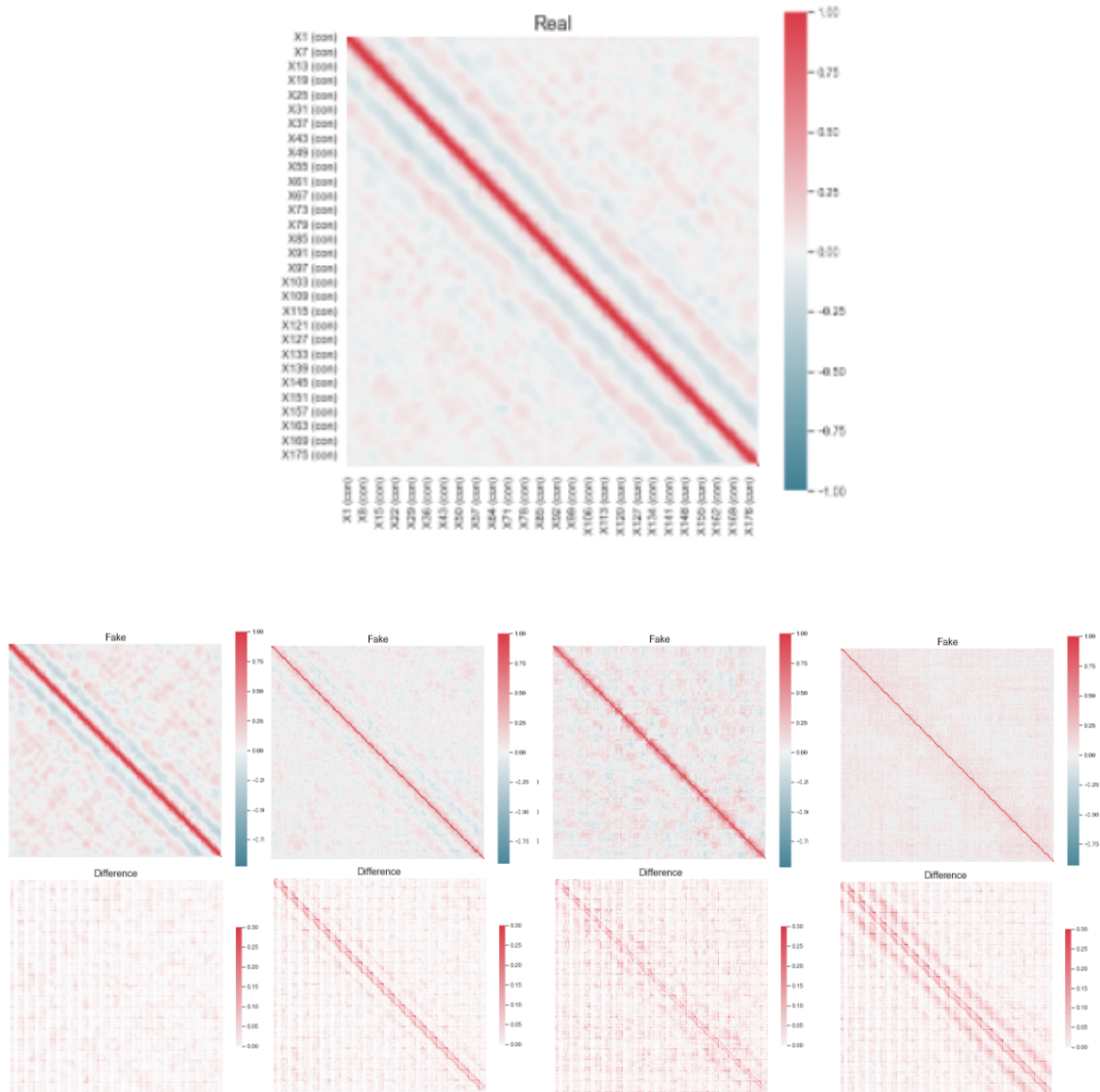


Figure D.2: The correlation matrices of real and synthetic datasets and their difference. Figures from left to right belongs to: WGAN, TGAN, CTGAN, CTABGAN

D.3 Column Distributions

D.3.1 Continuous Columns

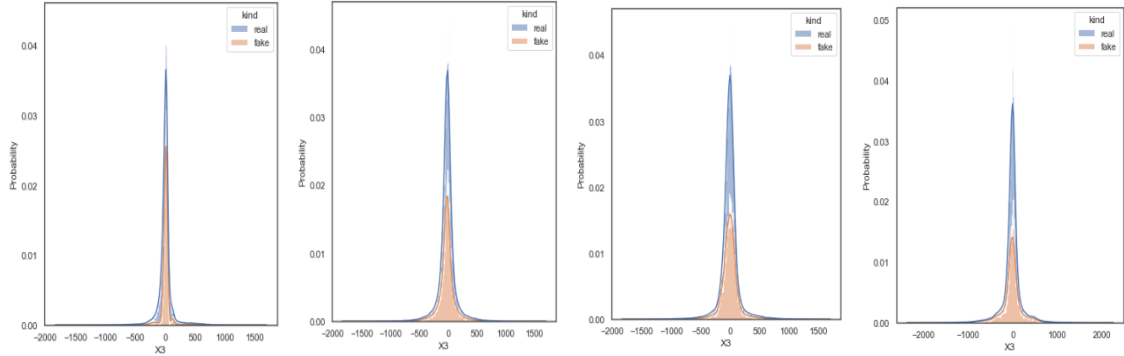


Figure D.3: The probability distributions of X3

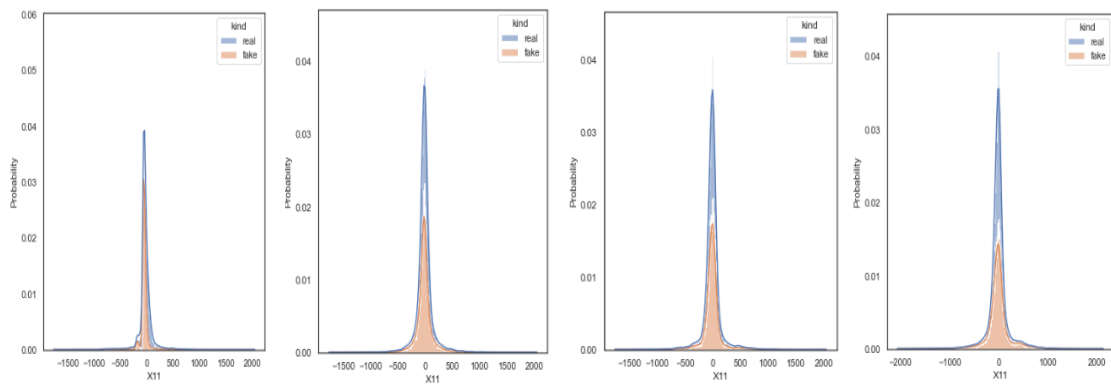


Figure D.4: The probability distributions of X11

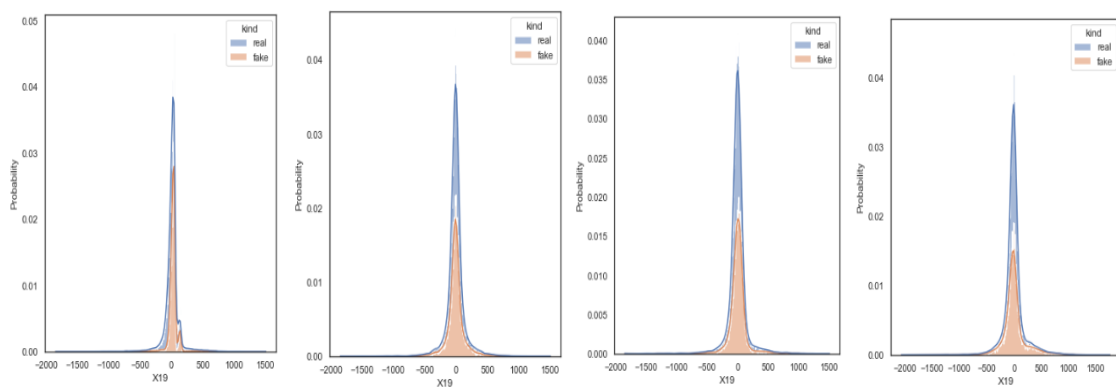


Figure D.5: The probability distributions of X19

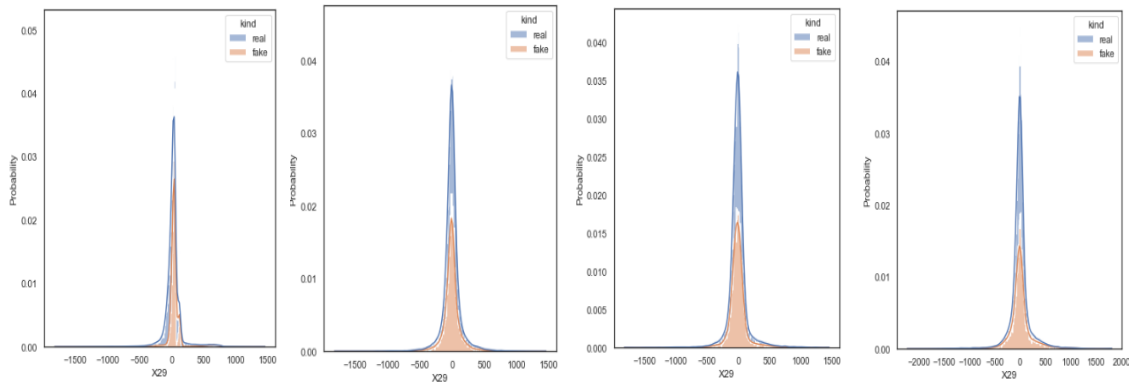


Figure D.6: The probability distributions of X29

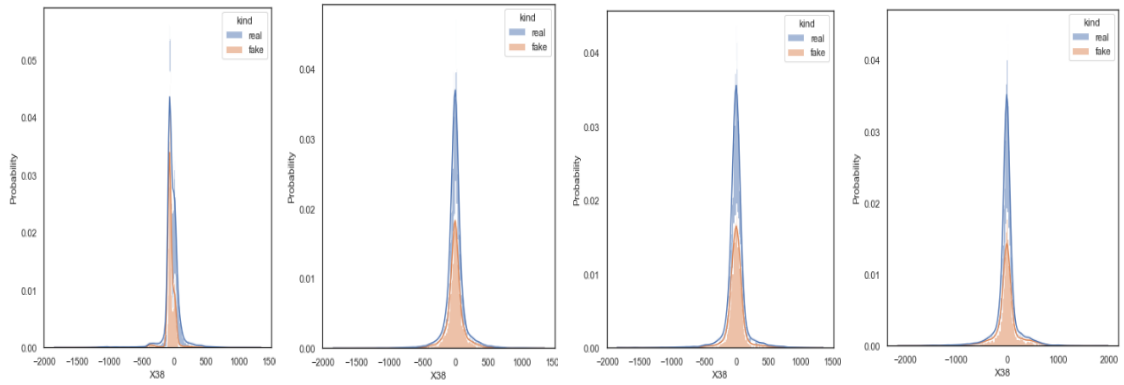


Figure D.7: The probability distributions of X38

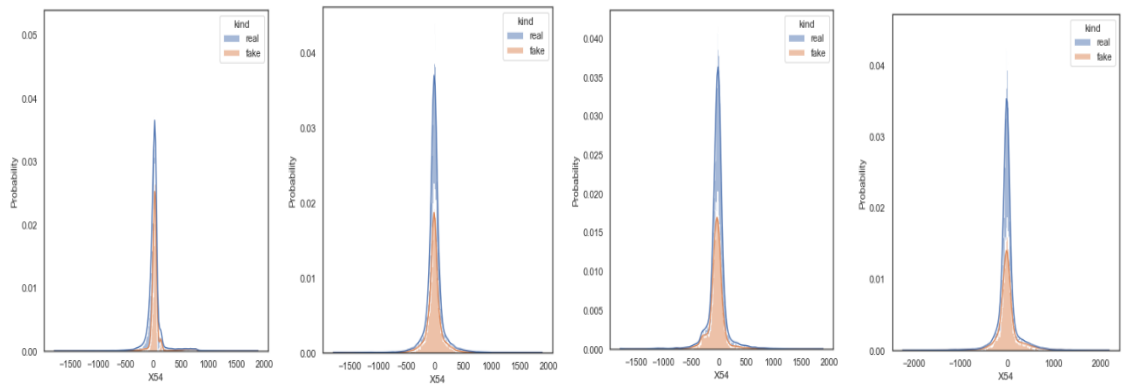


Figure D.8: The probability distributions of X54

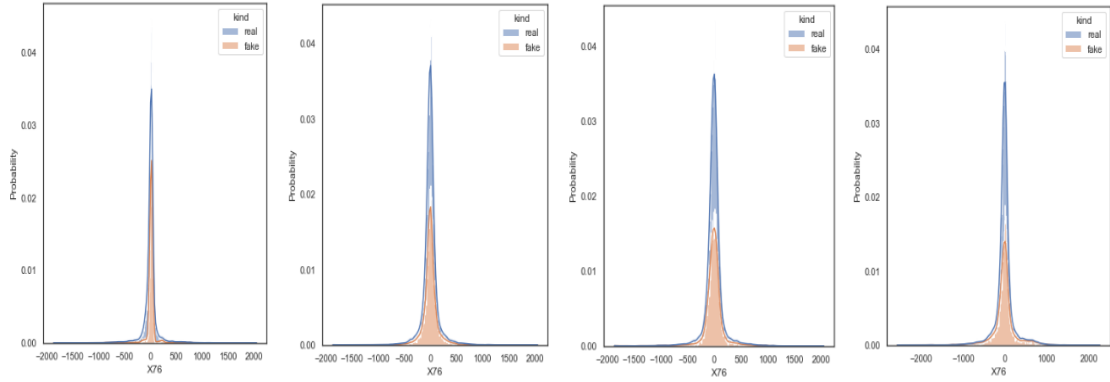


Figure D.9: The probability distributions of X76

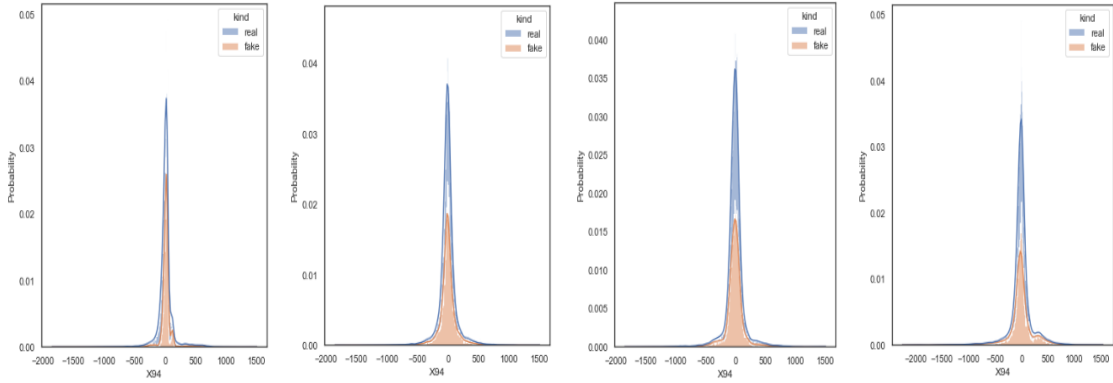


Figure D.10: The probability distributions of X94

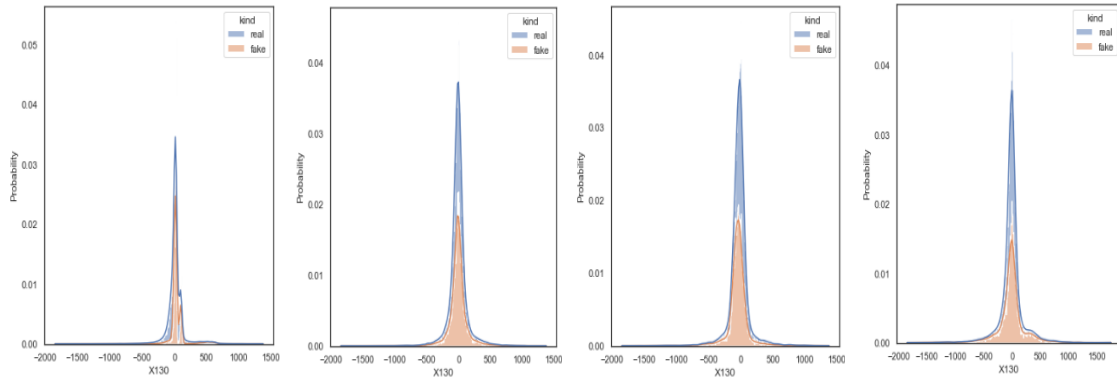


Figure D.11: The probability distributions of X130

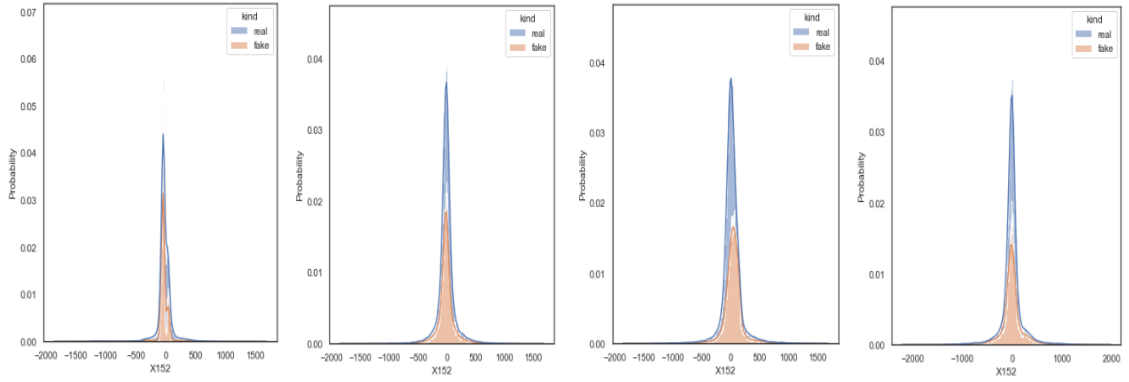


Figure D.12: The probability distributions of X152

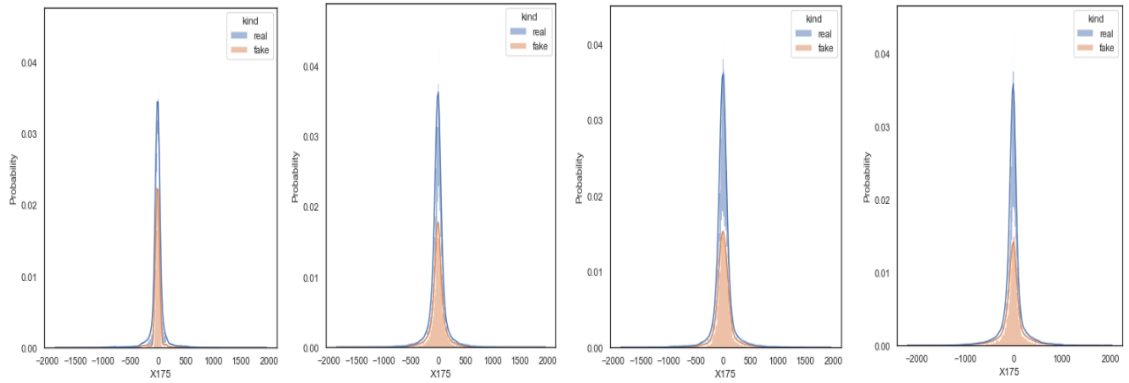


Figure D.13: The probability distributions of X175

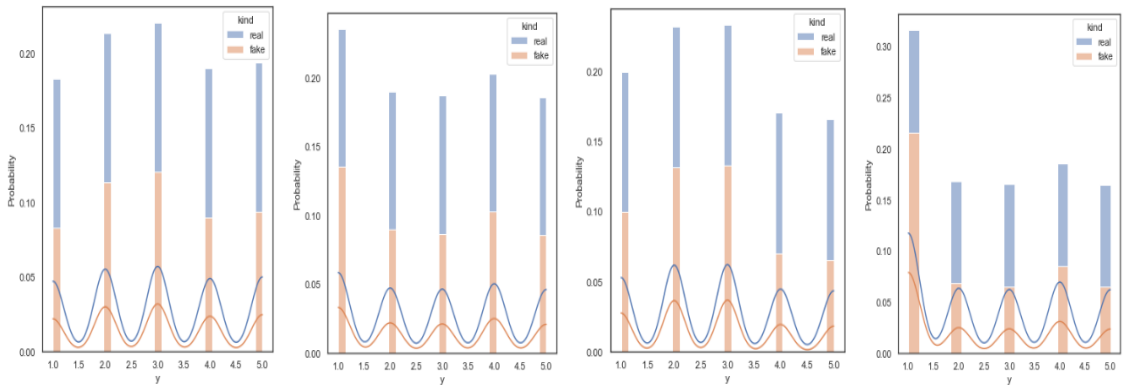


Figure D.14: The probability distributions of y

D.4 General Utility Scores

Table D.1: General utility scores. Columns from left to right represent: basic similarity score, Kolmogorov–Smirnov test score, Chi-squared test score, mean and standard deviation of the DCR distribution between synthetic and original records, correlation distance, Logistic Regression and SVM classification scores.

Model	Basic	KS test	CS test	DCR(μ)	DCR(σ)	Corr	LR	SVC
WGAN	0.875	0.948	NaN	7.55	8.45	6.9	0.77	0.62
TGAN	0.853	0.690	NaN	7.81	9.26	17.8	0.33	0.25
CTGAN	0.835	0.840	NaN	9.10	9.18	26.5	0.45	0.38
CTABGAN	0.868	0.909	NaN	8.08	9.27	19.8	0.73	0.59

D.5 ML Cross-Testing Scores

Table D.2: F1-scores of each classifier trained on the original and synthetic training datasets and evaluated on the real test data. DT, RF, LR and MLP stand for Decision Tree, Random Forest, Logistic Regression and Multi-layer Perceptron.

Classifiers	DT		RF		LR		MLP	
	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>	<i>real</i>	<i>fake</i>
WGAN	0.48	0.41	0.56	0.51	0.24	0.25	0.57	0.36
TGAN	0.48	0.27	0.56	0.28	0.24	0.23	0.57	0.22
CTGAN	0.48	0.27	0.56	0.31	0.24	0.16	0.57	0.18
CTABGAN	0.48	0.30	0.56	0.33	0.24	0.21	0.57	0.29

Table D.3: Difference of F1-scores of each classifier trained on the original and synthetic datasets.

Model	Δ F1-DT	Δ F1-RF	Δ F1-LR	Δ F1-MLP
WGAN	0.07	0.05	0.01	0.21
TGAN	0.21	0.28	0.01	0.35
CTGAN	0.21	0.25	0.08	0.39
CTABGAN	0.18	0.23	0.03	0.28

