Exploring the Value of GANs for Synthetic Tabular Data Generation in Healthcare with a Focus on Data Quality, Augmentation, and Privacy

Maria Elinor Pedersen

Thesis submitted for the degree of
Master in Applied Computer and Information Technology - ACIT
(Data Science)
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

Spring 2023

Exploring the Value of GANs for Synthetic Tabular Data Generation in Healthcare with a Focus on Data Quality, Augmentation, and Privacy

Maria Elinor Pedersen

Exploring the Value of GANs for Synthetic Tabular Data Generation in Healthcare with a Focus on Data Quality, Augmentation, and Privacy

http://www.oslomet.no/

Printed: Oslo Metropolitan University — OsloMet

Abstract

Artificial Intelligence has demonstrated immense potential in healthcare-related applications, paving the way for advancements in diagnosis, treatment, and patient care. However, data protection laws and regulations present challenges that hinder the progress of development. Consequently, synthetic data has emerged as an increasingly popular research area. Synthetic data can serve as an anonymized and representative alternative to real data. While various methods exist for generating synthetic data, Generative Adversarial Networks (GANs) have demonstrated exceptional performance in this regard. This thesis focuses on utilizing GANs to generate synthetic tabular data, given that a significant portion of today's data is organized in tabular format. The primary objective is to evaluate the capabilities of GANs in generating synthetic tabular data specifically for healthcare applications.

Three diverse healthcare datasets of varying sizes and complexities were selected, and two GAN models, CTGAN and CopulaGAN, were employed to generate corresponding synthetic datasets. The value of the generated data was assessed in terms of resemblance to real data, applicability to machine learning classification tasks, and preservation of individual privacy. Commonly used metrics within synthetic tabular data generation evaluation were applied to gauge the performance of the generated datasets. Resemblance metrics were based on comparing distributions and correlations between real and synthetic data. A novel framework, "SynthEval," was developed to offer an extensive evaluation of both real and synthetic data concerning classifier performance. Additionally, the framework investigated the potential of improving classifier performance by augmenting real data with synthetic data. Furthermore, the privacy assessment involved measuring nearest neighbor distances between real and synthetic data and checking for exact matches.

The findings indicate that GAN models have the potential to generate data that exhibit comparable performance to real data, given that the training data used for the GAN model is of sufficient quantity and not of low quality. The results also indicate that GAN models can generate cleaner data with less noise than real data. However, the study also reveals that when synthetic data performance is too high, it may result in compromised privacy.

Acknowledgments

I am grateful to my supervisors, Hårek Haugerud, Martin Gorosito, and Anis Yazidi, for their guidance and support throughout this project. Their expertise and feedback were instrumental in shaping my research, and I could not have accomplished this without them.

Also, special thanks to Kristine Løkke for her invaluable feedback and Olav Engelschiøn for his unwavering support, insightful discussions, and motivation. Lastly, I am also very grateful to my friends and family for their constant encouragement and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Artificial Intelligence (AI) and Machine Learning (ML) have shown tremendous promise in the healthcare industry. However, their full integration and realization have been hindered by various limitations. Despite their immense potential to revolutionize healthcare, challenges related to obtaining real medical datasets, complying with data protection laws and regulations, and limited interoperability between healthcare systems have impeded their widespread adoption. As a result, obtaining patient data for research purposes can be a time-consuming and challenging process, with approval times ranging from several months to years [2]. Consequently, researchers may have to rely on imbalanced and non-representative datasets, which can introduce bias and lead to inaccurate research results.

As a response to these challenges, researchers have turned to Synthetic Data Generation (SDG) using neural networks as a viable alternative [3]. SDG involves creating synthetic datasets that mimic real-world data, enabling researchers to overcome the limitations of obtaining real medical data. By utilizing SDG, researchers can generate diverse and balanced datasets, thereby enhancing the development and application of AI in areas such as diagnosis and treatment, patient engagement and adherence, and administrative tasks in the healthcare industry [4].

Synthetic data has emerged as a valuable tool in medical research due to its potential to create replicated and anonymous data, enabling researchers to experiment with more freedom while staying within legal boundaries [5, 6]. Ideally, synthetic data should imitate the underlying statistical properties of the real data without including any data from the original dataset. Its applications in medical research are diverse, including augmenting datasets that have few data points to improve machine learning models [7, 8]. This addresses the issue of data scarcity, a common challenge in medical research, by synthesizing additional data points that are similar to the existing ones. This increases the sample size and improves the accuracy of models, which rely on learning patterns from vast amounts of data and diverse and representative data, to achieve high performance. Synthetic data also has applications in privacy preservation, where the focus is on ensuring that data is safe to share among researchers without compromising privacy [6]. Synthetic data can be used to generate data that is similar to the original data, but without revealing any personal

information, making it safe for sharing and analysis.

Numerous methods have been researched to address the challenges of data scarcity and privacy concerns in medical research. One common approach involves using baseline models to replace or delete sensitive values or add noise to the data [9, 10, 11]. Statistical and probabilistic models can also be used to simulate the real data [12, 13, 14]. Although these methods offer promise, they may not be the most effective means available for generating high-quality data. Moreover, Deep Learning (DL) generative techniques have shown remarkable potential in generating highly realistic synthetic data. Autoencoders, Generative Adversarial Networks (GANs), and Ensembles are among the most researched approaches, with GANs being the most popular method, especially in the generation of images and videos [15]. GANs have seen significant improvements in the area of AI-generated images, including their application in medical research to generate realistic images of skin lesions [16], synthesize chest X-rays for COVID-19 detection [17], and create colonoscopic images [18]. However, the focus of this thesis is on a relatively novel and less-explored area of synthetic data generation, namely Synthetic Tabular Data Generation (STDG) using GANs.

While there is extensive research on SDG using medical images, the most popular method for developing machine learning models involves utilizing structured tabular data. As a result, Electronic Health Records (EHRs) and other health-related data hold significant potential for creating AI-based models. Nevertheless, there is a lack of progress in this field as structured tabular data is also more identifiable compared to other types of data [15]. Additionally, due to the STDG being a relatively new research area, there is currently no consensus on best practices for creating synthetic tabular medical data [19]. In this thesis, the overarching goal is to explore various methods and metrics used in this emerging field. Additionally, the aim is to contribute to the advancement of STDG research by implementing and testing several models on synthetic data. By investigating different approaches and evaluating their effectiveness using established metrics, this research seeks to expand the understanding of STDG techniques and their potential applications in healthcare.

1.2 Problem Statement

This thesis is a collaborative effort with the Centre for Intelligent Musculoskeletal Health (CIM) at OsloMet's Faculty of Health Sciences. In conjunction with the university's AI Lab, the research aims to evaluate the effectiveness of different GAN models for generating synthetic tabular data in medical research applications. The research will assess their performance using different metrics, including similarity, machine learning utilities, and privacy. This thesis will address three key problem statements:

1. How effectively can entirely synthetic datasets be generated to maintain predictive power comparable to real data?
2. How significantly can the performance of prediction models be enhanced by augmenting real data with synthetic data, as compared to using real data alone?
3. To what extent can privacy and data quality be balanced when generating synthetic medical data that accurately captures clinical complexity?

1.3 Thesis outline

Chapter 2 offers a comprehensive overview of the theoretical background required for the thesis. Topics such as Artificial Neural Networks, Generative Adversarial Networks, and various GAN architectures are covered. The chapter also delves into the process of generating synthetic tabular data using GANs and explores methods for evaluating synthetic data within related research.

Chapter 3 details the approach adopted for this thesis by first introducing the datasets and GAN models used for the study. Three different datasets were compared, and two GAN models, namely CopulaGAN and CTGAN, were used. This chapter then delves into how the models were implemented and how the generated data was evaluated.

Chapter 4 presents the outcomes derived from the model training and evaluation framework established for this thesis. The chapter offers a comparative analysis of the different datasets generated by the models, in three different evaluation dimensions.

Chapter 5 discusses the findings of the research in detail and highlights the limitations of the study. The chapter also outlines possible areas for future research.

Finally, Chapter 6 concludes the thesis by summarizing the research and highlighting the key contributions made by the study.

Chapter 2

Background and Related Work

This chapter provides a comprehensive review of the theoretical background required for this thesis. To gain a thorough understanding of Generative Adversarial Networks (GANs), it is essential to first understand the fundamental concepts of neural networks. The chapter then delves into the architecture of GAN, including commonly used variants. Finally, the chapter explores the application of GANs in synthesizing tabular data, covering different tabular GAN models and evaluation metrics.

2.1 Artificial Neural Networks

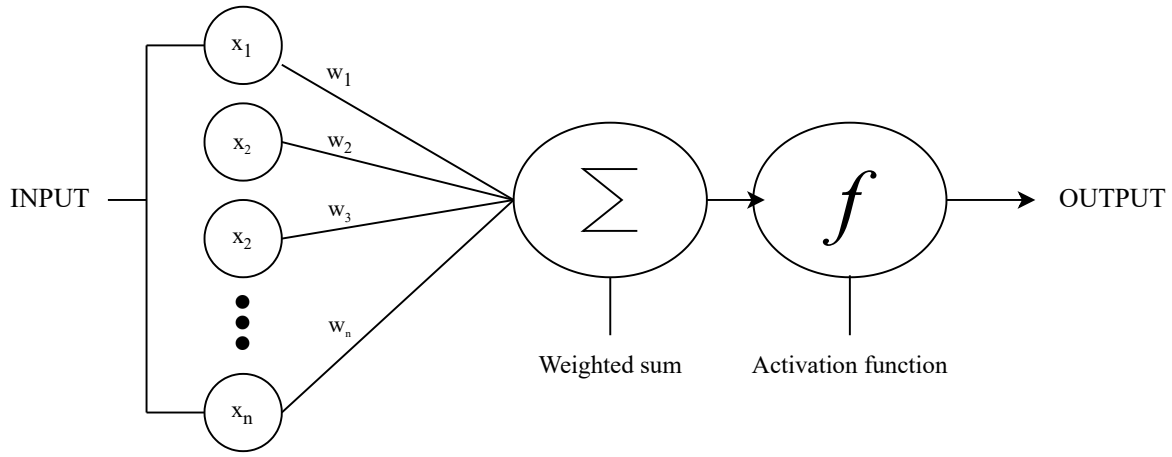


Figure 2.1: Simple artificial neuron architecture.

Artificial neural networks (ANNs), commonly referred to as neural networks, are a sub-field of Artificial Intelligence (AI) that models information processing similar to the human brain. The human brain performs computations through its intricate network of interconnected neurons (or nodes) that communicate by transmitting electrical impulses. A neuron computes an output based on weighted input received from other neurons. The relative importance of the input is determined based on the associated weight [20].

In an artificial neuron, input signals are weighted and summed, and the resulting value is compared to a threshold using an activation function, also referred to as a

step function. If the sum exceeds the threshold, the output is activated, otherwise, it remains inactive. The process of learning in an artificial neuron involves modifying the weights and threshold values to minimize the discrepancy between the predicted and actual output [21]. A simple artificial neuron architecture, shown in Figure 2.1, illustrates the fundamental behavior of a neuron and serves as the foundation for a neural network.

2.1.1 Feed-forward Neural Networks

A feed-forward network is a common neural network architecture that can be implemented with either single-layer or multi-layer functionality. A single-layer feed-forward network is composed solely of an input layer and an output layer. The input neurons receive the input data, while the output neurons provide the output results. The computations in a single-layer network are performed by the output layer, and the network has a unidirectional connection from input to output neurons due to its acyclic structure. Figure 2.2 provides a visual representation of this architecture.

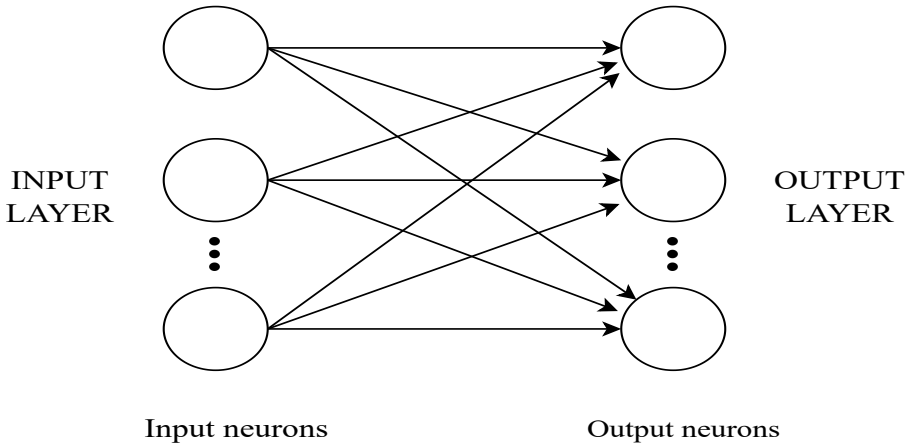


Figure 2.2: Single-layer feed-forward network.

A multi-layer feed-forward network operates similarly but with the addition of hidden layers between the input and output layers. These hidden layers assist the network in conducting intermediate computations prior to producing the final output. The computations carried out by the hidden layers may involve operations such as data transformation, automatic feature creation, and so on. The hidden layers are also unidirectional, and they transfer the training data from one layer to the next during the feed-forward process. The number of hidden layers required in a neural network is dependent on the complexity of the problem being addressed. While some problems may be adequately solved using a single hidden layer, others may require multiple ones [22]. Selecting an inappropriate number of hidden layers or neurons can lead to issues such as underfitting or overfitting. Further details on these concepts will be discussed in Section 2.1.4.

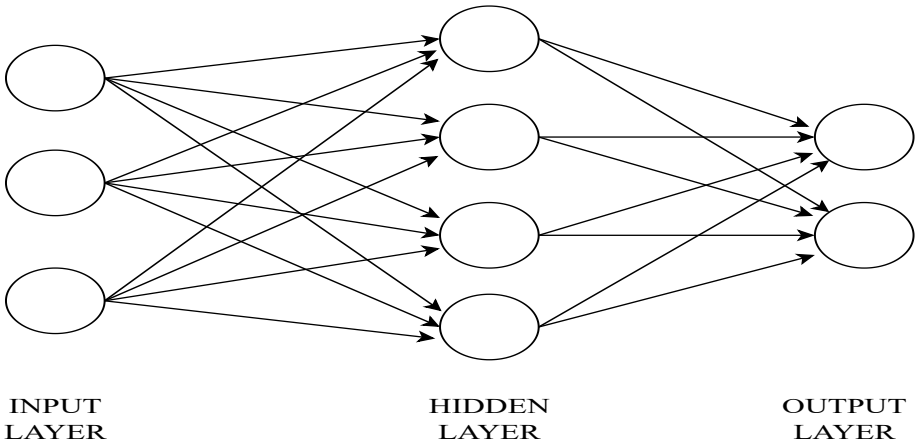


Figure 2.3: Multi-layer feed-forward network.

2.1.2 Training Neural Networks

Deep neural networks, which consist of multiple hidden layers, employ the backpropagation algorithm to facilitate the learning process. The algorithm iteratively adjusts the weights and biases of the neurons in the network to minimize the prediction errors. In simple terms, the backpropagation algorithm involves a forward pass where inputs traverse the network, and predictions are generated. The output is compared with the actual output, and an error is computed based on a loss function. The error is then propagated backward through the network, and an optimization algorithm modifies the weights. During the backward pass, the weights and biases are updated in a way that minimizes the loss by computing the gradient of the loss function with respect to them using the chain rule. This iterative process continues until the error is reduced to an acceptable level [21, 23].

Using an optimization algorithm is essential for finding the optimal set of weights and biases that minimize the error during neural network training. Therefore, selecting the best optimizer is critical to achieving an effective training process. While there are several optimization algorithms available, the focus here will be on explaining two widely used algorithms in modern machine learning: Stochastic Gradient Descent (SGD) and Adam.

Gradient Descent

The optimization of a neural network is most commonly achieved using gradient descent algorithms that attempt to minimize the cost function towards a local minimum. This learning algorithm has several variations, including the use of batch training, stochastic training, or mini-batch training. Batch Gradient Descent (BGD), commonly known as Vanilla Gradient Descent, is a fundamental optimization algorithm. It calculates the gradient of the cost function with respect to the parameters, θ, using the entire training dataset in each iteration. The goal is to find the optimal parameters, that will result in the lowest possible loss. The equation that updates each parameter is denoted as the following equation [24]:

$$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta) \tag{2.1}$$

The learning rate, η is the parameter that determines the step size. The step size then determines the speed of how quickly the model converges. While a larger step size can speed up the training of a model, it also increases the risk of overshooting the optimal value. On the other hand, a smaller step size can result in a more precise convergence but at the cost of a longer training time. The gradient of the cost function with respect to the parameters, denoted as $\bigtriangledown_\theta J(\theta)$, is a crucial component in the optimization process. The algorithm leverages this gradient to iteratively adjust the parameters in an attempt to minimize the cost function. Lastly, the cost function, denoted as $J(\theta)$, quantifies the model's loss. Since this algorithm uses the entire dataset to compute the gradients, it can be slow and computationally expensive on large datasets [24]. On the other hand, the Stochastic Gradient Descent (SGD), updates each parameter for each training example. SGD is calculated by the following equation:

$$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta; x^i : y^i) \tag{2.2}$$

Within the function, x^i represents each training example in the dataset, while y^i corresponds to its respective label. This approach is more computationally efficient then BGD as it avoids the need to compute gradients for the entire training dataset in each iteration. Instead, it randomly selects a single training example and updates the gradient based on that. SDG, due to its inherent variability, has the potential to reach local minima quicker than BDG. However, it also risks overshooting the exact minimum, and therefore hinder convergence [24]. Moreover, a compromise between these two approaches is Mini-batch Gradient Descent, denoted in the following equation:

$$\theta = \theta - \eta \cdot \bigtriangledown_\theta J(\theta; x^{i:i+n} : y^{i:i+n}) \tag{2.3}$$

Using this equation, the gradients are updated in small batches of n training examples. This results in a balance between computational efficiency and a more stable convergence towards a minimum of the cost function. Hence, mini-batch is often the preferred choice among the gradient descent algorithms when training a neural network [24].

Adam

Adam is short for Adaptive Moment Estimation, and is an optimization method that combines the two methods; SGD and momentum. Specifically, the Adam uses the advantages of AdaGrad [25] and RMSProp [26] which is two popular optimization algorithm, to optimize neural network training with little memory requirement.

What makes this algorithm unique is that it computes adaptive learning rates for each parameter during training, based on estimates of the gradient's first and second moments. The algorithm allows for dynamic adjustments of the learning rate, which are based on estimates of the past gradient and its variations over time [27]. In contrast, SGD employs a fixed learning rate for all the gradient updates, without any changes during training. Adam's adaptability allows for efficient optimization of neural networks, even in complex parameter spaces, making it a key advantage of this algorithm [27].

2.1.3 Activation Functions

As previously stated, neural networks also include nonlinear activation functions. Activation functions play a crucial role in neural networks as as they introduce nonlinearity, allowing for more complex and powerful modeling of data [28]. Some commonly used activation functions include the Sigmoid function, Hyperbolic Tangent (tanh), and Rectified Linear Unit (ReLU). The Sigmoid function is a common, but more simple, activation function in neural networks. Its definition is as follows:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.4}$$

The functions take an input value x, and maps the value to range between 0 and 1. This function is commonly used in the output layer of shallow neural networks for binary classification problems [28]. However, the function has some issues, such as the vanishing gradient problem that arises in networks with 3-4 hidden layers. Hence, other activation functions are often preferred. One such function is Tanh, which is an improved version of Sigmoid, and is defined by the ratio of the functions of sinus and cosine:

$$tanh(x) = \frac{\sin(x)}{\cos(x)} \tag{2.5}$$

It can also be abstracted from the sigmoid function, where *sigmoid* is g(x) from Equation 2.4:

$$tanh(x) = 2sigmoid(2x) - 1 \tag{2.6}$$

This function maps the input value to a range of -1 to 1, making it faster and more accurate than the sigmoid function. It also has a wider range of output values, allowing it to better capture negative and positive values and handle data that requires a wider distribution [28]. Additionally, the Rectified Linear Unit (ReLU) function is another popular activation function. It is defined as follows:

$$g'(x) = max(0, x) \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{2.7}$$

Again, x is the input function and the values that the function outputs are the maximum of 0 and x. Meaning that values that are greater or equal to 0, output x, while values less than 0, output 0. The function is popular due to its ability to prevent vanishing gradients and its computational efficiency. Another variant of ReLU is Leaky ReLU, which addresses the issue of "dying ReLU" where some neurons become permanently inactive during training [28].

2.1.4 Regularization Techniques

The performance of a neural network is significantly influenced by both the quality and quantity of training data. If the training data is insufficient, two common issues can arise: overfitting and underfitting. Overfitting occurs when a model performs well on the training data, but its performance degrades significantly when tested on unseen

data. A common indication of this is a low training error but a high validation error. This suggests that the model is memorizing the expected outputs instead of learning the underlying data distribution. On the other hand, underfitting is a problem where the neural network cannot learn enough from the limited data available, resulting in poor performance. High training and validation errors are common indicators of this [29]. To prevent these issues from happening, neural networks use regularization techniques. Regularization is when small variations are added to the data, to train a model more efficiently. Common regularization techniques include L1 and L2 regularization, dropout, and batch normalization.

L1 regularization, also known as LASSO regularization, adds a penalty to the loss function based on the absolute values of the weights when training a neural network. The goal of L1 regularization is to eliminate irrelevant features from the model by promoting sparse feature selection and improving generalization performance. This regularization technique encourages the neural network to select the most important features, making the model less complex [30]. Furthermore, L2 Regularization, commonly known as ridge regularization or Tikhonov's regularization as well, also penalizes the weights in the loss function. However, the penalty is added based on the squared values of the weights [31]. Therefore, L2 is less aggressive when it comes to feature selection and should be used when all features are considered equally important.

Another commonly used regularization technique in neural networks is dropout, which randomly drops nodes in the network using a dropout layer. When a node is dropped, its information connected to the incoming and outgoing connections is ignored. The purpose of dropout is to encourage each node in the network to learn information independently, without overly relying on its connections. In turn, this makes the network more resilient to noise and variations in input data [31].

During the training of a neural network, the distributions of the layer inputs tend to change between each layer, which is commonly referred to as internal covariance shift. This shift slows down training because it requires lower learning rates and more thorough optimization of parameters. Batch normalization addresses this concern by normalizing the inputs to each layer during training, for each mini-batch, to ensure that the network maintains consistent layer input distributions. As a result, batch normalization helps to enhance the generalization performance and stability of the model [32].

2.2 Generative Adversarial Networks

The most prevalent approach used in machine learning today is supervised learning. In this approach, algorithms learn from labeled data, where the model is exposed to a set of inputs and their corresponding outputs. It then learns to map the data through training [33]. This is commonly used with classification and prediction tasks. Despite its capability of delivering higher accuracy than human counterparts, it still depends on human-supplied labeled data. As a result, many researchers are now exploring unsupervised learning to reduce the dependence on human supervision and the amount of training data necessary.

Generative modeling is a type of unsupervised learning, where a set of samples is taken from an unknown data-generating distribution P_{real}, and the goal is to find an

estimate P_{model} that is similar to it. A well-known variant of generative modeling is the Generative Adversarial Network (GAN) [34]. GANs have a wide range of applications, but as of now, their primary strength lies in the synthesis of images.

2.2.1 Structure and Training

A GAN is a type of neural network architecture that involves a pair of neural networks competing against each other to learn data patterns. This method can both be semi-supervised and unsupervised [35]. How this works, is depicted in Figure 2.4.

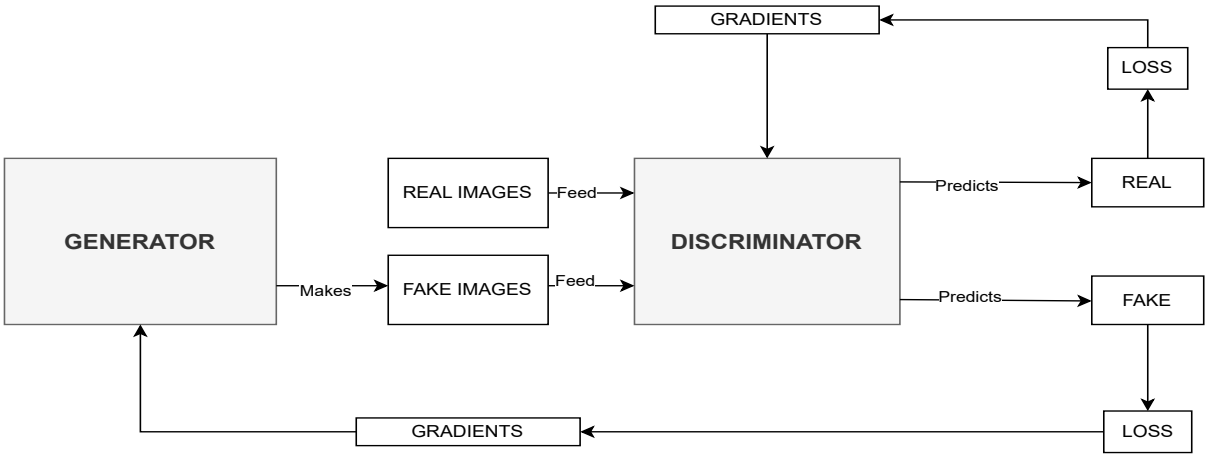


Figure 2.4: Overall GAN structure.

GANs consist of two components: a generator and a discriminator. The generator produces counterfeit images that are as authentic-looking as possible, while the discriminator is presented with both real and fake images and attempts to differentiate between them. The generator is unable to access real images but improves through its interactions with the discriminator. However, the discriminator has access to both the synthetic images and samples from the real images. The discriminator evaluates error by comparing the ground truth of the image with its prediction, using backpropagation. This error is also fed to the generator which attempts to improve itself by generating more realistic and high-quality images [35, 34]. The objective is to achieve a balance where P_{real} and P_{model} are nearly equal.

During the training of a GAN, both of the neural networks involved are simultaneously optimized. The objective is to enhance the classification accuracy of the discriminator by adjusting its parameters while also modifying the generator's parameters to successfully deceive the discriminator. When one network updates its parameters, the parameters of the other network remain fixed.

A value function $V(G, D)$ for a simple GAN is used to evaluate the cost function when training. Training involves solving the following minmax function [36]:

$$\max_D \min_G V(G, D),$$

Where,

$$V(G, D) = \mathbb{E}_{p_{data}(x)} log D(x) + \mathbb{E}_{p_z(z)} log(1 - G(z)) \tag{2.8}$$

The research paper by Goodfellow et al. [36] provides a more comprehensive proof of this equation. However, simply put, G and D represent the generator and discriminator functions, respectively. The probability distribution of the latent space is denoted by p_z, while the probability distribution of the training dataset is denoted by p_{data}. When a data sample, x, is drawn from p_{data}, the discriminator aims to classify it as a real sample. On the other hand, when a generated sample, $G(z)$, is given as input to the discriminator, it aims to classify it as a fake sample. The variable z represents a random noise vector drawn from the latent space probability distribution, p_z. G wants to minimize V, while the discriminator D wants to maximize it.

The discriminator is optimal when:

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \tag{2.9}$$

While the generator is optimal when:

$$p_g(x) = p_{data}(x) \tag{2.10}$$

This implies that when the accuracy of the discriminator reaches 50% for all the samples taken from x, the generator is considered optimal. In simpler terms, the discriminator is confused because its predictions of whether the data is real or fake are as good as a random guess.

2.2.2 Challenges with GANs

Despite the existence of multiple theoretical approaches to training, GANs are well-known for being challenging and unstable to train. Common failures that can happen during training are the failure to converge, vanishing gradients, mode collapsing, hyperparameter tuning and oscillatory loss [37, 38]. These challenges are general problems that can arise when working with GANs, and are not dependent on a specific domain or architecture.

When a machine learning model reaches convergence, it means that the model ceases to improve with further training. This is desirable as the algorithm has succeeded in finding the optimal solution to the problem. Considering that GANs train two models at the same time, it is difficult to make the models converge. In GANs, low-quality generated images can serve as a sign of non-convergence. Non-convergence often occurs when either the discriminator or the generator becomes more powerful, resulting in a lack of learning and a potential issue with vanishing gradients [37]. Poor initial generated images increase the discriminator's ability to distinguish between real and fake images. Consequently, the probability of the generated images being real will be close to zero. If this continues, the discriminator will cease to supply gradients to the generator, hindering its ability to improve [37].

GANs have a tendency to produce output that lacks diversity and only captures a small portion of the data distribution. This limitation typically stems from the generator becoming trapped in suboptimal solutions, leading to generating outputs that do not fully reflect the target data. Another cause can be an overly strong discriminator, making it difficult for the generator to find a way to trick it [39]. Additionally, without proper regularization, mode dropping can also play a role [40].

When the target distribution has multiple clusters of similar data points, the generator may only generate data that belongs to a certain mode of class, leading to a lack of diversity in the generated dataset.

Oscillatory loss may occur during the training process, which can be recognized by unstable and wild oscillations in the discriminator or generator loss. The loss function should ideally stabilize or gradually decrease/increase over time. However, if it does not, it can hinder the effectiveness of the training process [38]. Furthermore, the proper selection of hyperparameters is crucial for achieving convergence in GANs. However, due to the large number of hyperparameters involved, finding the optimal settings can be a challenging task. While Grid Search is a common technique used to tune hyperparameters in neural networks, it may not be feasible for GANs due to the computational cost and time required, especially when dealing with a large number of parameters. This can make it hard to find the optimal parameters for the network [38].

2.3 Different GAN Architectures

In 2.2, the most basic structure of a GAN was presented, commonly known as the Vanilla GAN or Traditional GAN. This GAN concept and structure was first introduced by Goodfellow et al. [36] in 2014, and over time, several variations of its architecture have been developed. Within the context of this thesis, this section will provide an overview of some of the most fundamental GAN architectures.

2.3.1 Conditional GAN

Conditional Generative Adversarial Networks (CGANs) were first proposed by Mirza and Osindero [41] in 2014, aiming to address the limitation of traditional GANs that lacked control over the generated output. Unlike traditional GANs that rely solely on random noise input, CGANs incorporate additional conditional information during the training process to have more control over the generated data. The use of conditional inputs, such as class labels or data from other modalities, is an effective way to enhance the capabilities of GANs. By incorporating such inputs, CGANs allow for more precise and controlled data generation. During training, CGANs can be conditioned on a specific class, allowing the network to generate images that belong to that class. This is in contrast to traditional GANs, which lack this ability, and as a result, it is often challenging to generate data samples that belong to a specific class.

The traditional GAN is modified by incorporating extra conditional information, denoted as y, into both the discriminator and generator. This conditioning can be achieved by introducing an additional input layer that takes in the conditional information. As a result, Equation 2.8, which represents how a traditional GAN trains, can be updated to include y as an input to both the discriminator and generator, allowing for the conditioning to take effect during the training process. The updated equation can be observed in Equation 2.11.

$$V(G,D) = \mathbb{E}_{p_{data}(x)} log D(x|y) + \mathbb{E}_{p_z(z)} log(1 - G(z|y)) \tag{2.11}$$

This GAN network has laid the groundwork for a lot of applications and newer GAN architectures today. Examples of tasks this architecture can be used for are image-to-

image translation [42], convolutional face generation [43], shadow detection [44], and other tasks where conditional information provides control over the generated output.

2.3.2 Deep Convolutional GAN

Deep Convolutional Generative Adversarial Networks or DCGANs were created to improve upon the traditional GAN architecture for image generation tasks. Radford et al. [45] found that traditional GANs had issues with generating high-quality images and offered some architectural changes in a paper published in 2015. The main architectural change that the authors introduced was to include deep Convolutional Neural Networks (CNNs) in the discriminator and the generator.

Simply explained, CNNs are a type of deep-learning neural network that has been proven to work great with image recognition, object localization, and image segmentation tasks. The way that the network operates is to identify patterns in input images through several convolution and pooling layers. Each layer tries to extract different features from the image and ultimately it leads to a prediction or classification. The convolutional layer uses filters on the image to detect different features, such as edges or shapes. While the pooling layer downsamples the output from the convolutional layer to reduce the complexity of the network [46].

While DCGANs has shown to be more effective at generating high-quality images compared to traditional GANs, the training process of these networks remains challenging. Radford et al. [45] addressed these challenges by introducing a couple of architecture guidelines to help stabilize the training of the network. Firstly, the paper proposed replacing any of the pooling layers in the discriminator with strided convolutions, while fractional-strided convolutions should be used in the generator. Additionally, batch normalization should be used in both the generator and discriminator. Batch normalization stabilizes learning by normalizing the input to the model. For deeper network architectures, it is recommended to remove fully connected hidden layers. Lastly, ReLU activation functions should be used in all the layers of the generator, except for the output layer which should use the Tanh activation function. While the discriminator should use the LeakyReLU activation function in all layers.

2.3.3 Wasserstein GAN

The Wasserstein Generative Adversarial Network (WGAN) was introduced in 2017 by Arjovsky et al. [47] and this architecture focused more on training stability and addressed the issue of mode collapse, where the generator produces limited and repetitive samples. It also focuses on the generation of a more meaningful loss metric, resulting in more informative learning curves. This can be a valuable tool for debugging and optimizing hyperparameters, as it helps to better understand the training dynamics and performance of the model.

What distinguishes this architecture from a conventional GAN is that the discriminator does not rely on binary classification to determine whether an image is real or fake. Instead, it employs the Earth Mover's Distance (EMD), also known as the Wasserstein metric. The discriminator, or "critic" as Arjovsky et al. [47] calls it, utilizes the EMD as a loss function. The EMD is a way to measure how different the real and generated data distributions are, based on how the discriminator function is set up. The critic tries to predict this distance by examining different mappings of the real

and synthetically generated data into a shared space. It then aims to make it as challenging as possible for the generator to deceive it. This is done by trying to maximize the amount of probability mass that needs to be moved. The generator's goal is then to minimize the EMD between the expectations of the real and generated distributions [48, p.83-88]. WGANs also introduced the 1-Lipschitz into the discriminator function. This is a technical constraint that aims to make the network more stable to train.

EMD makes WGAN's loss more understandable since it does not involve any logarithms. It also makes hyperparameter tuning easier, because WGAN requires a clipping constraint during training which acts like a typical learning rate. This in turn gives clearer stopping criteria when training, because the network can just calculate the Wasserstein distance to give an indication of when to stop to reach convergence [48, p.83-88]. There has also been done some research on whether traditional measures of the training process of GANs actually are meaningful [49]. This is training measures such as Jensen-Shannon (JS) loss and divergence between the generator and the real distribution. Therefore, with WGAN utilizing Wasserstein distance, it manages to overcome this.

In the original WGAN article, the authors did not explicitly design the network to address mode collapse, but they claimed that the new distance metric used in WGAN could alleviate this problem. This is due to the EMD-based critic utilized in WGAN being less prone to vanishing gradients and overfitting. Because of this, the critic manages to provide the generator with a lot more informative feedback, resulting in more diverse output samples that capture the different modes of the target distribution. This, in turn, helps to alleviate the problem of mode collapse during GAN training.

2.3.4 Wasserstein GAN with Gradient Penalty

The original WGAN architecture was a significant step towards achieving more stable training in GANs. However, it still faced challenges such as generating low-quality samples and issues with network convergence. To address these limitations, Gulrajani et al. [50] proposed an improved network architecture called WGAN-GP in their publication. WGAN-GP introduced a gradient penalty to the architecture, aiming to alleviate these challenges and improve the overall performance of the GAN model. The authors' key observation was that the limitations in the original WGAN were likely due to the enforcement of the Lipschitz constraint on the critic's weight clipping, resulting in undesirable behavior. The Lipschitz constraint used in WGAN is a constraint on the smoothness of the critic's output. As an alternative approach, they proposed a method that penalizes large gradients, which results in more stable training and smoother gradients without the need for explicit weight clipping.

To control how strong the penalty should be, they applied a coefficient, λ, to the training process. A higher value of λ results in a stronger penalty, which can contribute to increased stability during training. The downside of this is that it can make the training more difficult. Gulrajani et al. found that a $\lambda = 10$ worked well across different architectures and datasets, however, other penalties may be better suited for other models as it all depends on the specific problem at hand.

The authors of the article also highlight that they deviate from the common practice of using batch normalization in both the discriminator and generator in many GAN

architectures. In their approach, they do not use batch normalization in the critic. Instead, they opt for layer normalization as a replacement. In conclusion, the use of WGAN-GP outperforms the original WGAN, without the need for much hyperparameter tuning, as demonstrated by Gulrajani et al.

2.4 Generating Tabular Data Using GANs

Although basic statistical models can generate synthetic tabular data, their scalability and effectiveness are limited, especially when dealing with larger datasets [51]. However, researchers have explored the potential of using GANs, commonly associated with image synthesis, for Synthetic Tabular Data Generation (STDG) as well [38]. When using GANs for STDG, the process is similar to image synthesis. The generator is trained to understand the data distribution across all the columns of the real dataset, allowing it to create a synthetic table that mimics the real one. In contrast to image data, tabular data can consist of various data types in each column. A column in a dataset may contain numerical values that are discrete or continuous, as well as categorical values. Consequently, the GAN employed for STDG needs to be able to learn and generate different types of data [52].

While the use of GANs in STDG is a relatively new research area, this section will cover the most prevalent tabular GAN approaches, the challenges they face, and how to evaluate the synthetic data generated.

2.4.1 Overview of Tabular GAN Approaches

Based on Hernandez et al. [15]'s review of 34 publications on STDG approaches, it was found that GAN-based methods performed better than other approaches tested. However, they did not identify a specific GAN-based approach that was the best. Similarly, Coutinho-Almeida et al. [3] reported in their review that none of the tested STDG approaches outperformed others across all dimensions. The challenge in comparing GAN-based approaches lies in the diverse data types, dimensions, and metrics used to evaluate model performance across different publications. As there is no universal GAN-based architecture suitable for all tabular data, the optimal GAN model architecture should vary depending on the dataset characteristics. For example, some models perform better with categorical and numerical data, while others are better with time-series data. Therefore, it is crucial to evaluate dataset features and experiment with different GAN-based architectures to find the most effective approach. Overall, achieving a balance between resemblance, privacy, and utility remains a challenging task as reported by various publications. As a result, there is a significant amount of work required to improve the generalizability of GAN models and identify the optimal model for all tabular healthcare data types [15]. With these considerations in mind, it is worth exploring a few noteworthy tabular GAN models that have gained attention and popularity.

Medical GAN (MedGAN), proposed by Choi et al. [53] in 2017, is an early example of a tabular GAN model and was designed specifically for generating synthetic medical data. MedGAN is able to generate high-dimensional, multi-label discrete variables that represent realistic patient data. The focus is mainly on binary and count variables. Additionally, MedGAN incorporates an autoencoder with the GAN framework to effectively learn the underlying distributions of discrete features in a dataset. It also

uses mini-batch averaging as a technique to address mode collapse and is built upon the traditional GAN architecture by Goodfellow et al. [36].

Although MedGAN has shown promising results, alternative architectures have been proposed that improve upon its design. Baowaly et al. [54] (2018) modified MedGAN with the aim of creating more realistic patient records. Two solutions, MedWGAN and MedBGAN, were proposed. MedWGAN replaced MedGAN's GAN model with WGAN-GP, while MedBGAN used a boundary-seeking GAN [55] instead. Both these models outperformed MedGAN in statistical and machine learning tasks, with medBGAN demonstrating better performance compared to both.

Camino et al. [56] (2018) proposed a method that was trained to generate multi-categorical data. The focus was to generate better discrete values, as previous GANs usually performed better with continuous data. The proposed method's architecture adapted MedGAN's decoder and incorporated multiple (Gumbel) softmax output layers that are suitable for categorical data structures. This method showed improvement over the original MedGAN architecture. However, its reliance on additional information, such as the dimensionality of the variables, limited its applicability as this information may not always be readily available. The HealthGAN architecture, proposed by Yale et al. [57] (2020), addressed limitations in MedGAN, such as its compatibility only with binary data and issues with resemblance comparison between real and synthetic data. HealthGAN integrated concepts from both the MedGAN architecture and WGAN-GP, so that it could make the model handle categorical features better.

In 2018, Park et al. [58] introduced TableGAN, an alternative GAN architecture that focuses on generating synthetic records with categorical, discrete, and continuous values. Based on the DCGAN model, TableGAN includes an additional classifier to ensure semantic integrity of the generated records. The classifier makes the model resilient against various attacks, such as re-identification attacks, attribute disclosure, and membership attacks.

Xu and Veeramachaneni [59] introduced Tabular GAN (TGAN) in 2018 and is an architecture that is similar to the TableGAN model. However, while TableGAN uses CNNs, Tabular GAN uses Recurrent Neural Networks (RNNs). TableGAN's main focus is on making a tubular GAN model that can handle various data types, including multinomial/discrete and continuous values. This model employs a form of RNN called Long-Short Term Memory (LSTM) network with an attention mechanism. This enables the model to generate values for each column sequentially, instead of generating the entire table at once.

The same authors who developed TGAN also introduced a refined model called CTGAN [60], which stands for Conditional Tabular GAN. This model improved upon TGAN, by making it possible to generate synthetic data with specific conditions or constraints. CTGAN addresses non-Gaussian and multimodal distributions in tabular data with mode-specific normalization techniques. It also uses a conditional generator and training-by-sampling to handle imbalanced discrete columns. The model is trained using WGAN-GP. A variation of CTGAN known as CopulaGAN is a model in the online Synthetic Data Vault (SDV) library [61] that utilizes the Cumulative Distribution Function (CDF) based transformation applied by GaussianCopulas. This

transformation makes it easier for the CTGAN model to learn the underlying data.

Traditional GANs are better at capturing static data samples, however the architecture has issues in capturing the unique temporal correlations that exist in time-series data. This is why, Yoon et al. [62] created a Time-series GAN (TimeGAN) that takes this into account. TimeGAN incorporates autoencoding components to complement its GAN components. Specifically, it includes an embedding function and a recovery function in addition to a sequence generator and a sequence discriminator. These components are trained together to enable the model to encode features, generate representations, and iterate across time. While the paper does not explicitly say what GAN architecture it is based on, the authors mention that Recurrent Conditional GAN (RCGAN) [63] and Continuous- RNN-GAN (C-RNN-GAN) are the closest in architecture [64].

Two more recent architectures that have gained attention in the field are C-TABGAN [65] and TabFairGAN [66], which were proposed in 2021 and 2022 respectively. C-TABGAN, or Conditional Tabular GAN, is a novel GAN architecture that enhances traditional tabular GANs by effectively modeling diverse data types, while also addressing data imbalance and long tail distributions. The model incorporates information, classification, and generator loss into the conditional GAN framework, building upon the CGAN architecture.

TabFairGAN is a recent approach that integrates fairness into the GAN architecture. The model tries to generate synthetic data as similar to the original dataset, while also making sure that the generated data is both accurate and fair. It does this by adding a fairness constraint in the value function. The underlying architecture is based on the WGAN framework.

Tabular GAN approach	**Year**	**Baseline Architecture**
MedGAN [53]	2017	Vanilla GAN, Auto-encoder
medWGAN [54]	2018	WGAN-GP
medBGAN [54]	2018	BGAN
MedGAN with Gumbel-softmax [56]	2018	Traditional GAN, Auto-encoder
HealthGAN [57]	2020	WGAN-GP
TableGAN [58]	2018	DCGAN
TGAN [59]	2018	Traditional GAN
CTGAN [60]	2019	WGAN-GP
CopulaGAN [61]	2019	WGAN-GP
TimeGAN [62]	2019	RCGAN, C-RNN-GAN
CTAB-GAN [65]	2021	CGAN
TabFairGAN [66]	2022	WGAN

Table 2.1: Overview of popular Tabular GAN approaches and the main architecture it is based on.

2.4.2 Challenges with Tabular GANs

GANs may face several difficulties when learning from tabular data. For example, imbalanced categorical data may lead to mode collapse when some categories have a significantly higher number of instances. This is especially an issue with Electronic Health Records (EHRs), as these often have issues with class imbalance [52]. Class

imbalance occurs when one or more classes have a significantly higher number of instances compared to others, typically referred to as majority classes. This imbalance can introduce bias in the model's performance and predictions.

Furthermore, the shape distribution of each column can differ, which can cause non-convergence and vanishing gradient problems [67]. Also, when the data is sparse and one-hot-encoded, it can cause problems during the discriminator training process. This is because the discriminator learns to differentiate between real and fake data based on how uncommon they are in the distribution, rather than based on whether they are real or fake values [67].

To overcome the challenges associated with learning from tabular data, researchers often use available datasets for research purposes. However, many open-source datasets lack standardized documentation and may have quality issues, posing challenges for data reliability and validity [19]. This is particularly problematic in the medical field, where class imbalance and data sparsity are common issues that can introduce bias in the model's performance and predictions. Additionally, open-source health-related datasets may have been previously anonymized and synthesized, leading to further bias. To address these challenges, it is advisable to use diverse datasets in various contexts, including authentic data directly sourced from hospitals or laboratories without any anonymization or modification.

2.4.3 Evaluation Metrics

To evaluate the performance of GAN-based approaches for STDG, three dimensions are commonly utilized: Resemblance, Machine Learning Utility, and Privacy. Each dimension has specific metrics that are used to measure the performance, as documented in relevant publications. Analyzing these dimensions and associated metrics provides a deeper understanding of how GAN-based approaches are assessed for their effectiveness in generating synthetic tabular data while preserving important characteristics such as resemblance to the original data, usefulness for machine learning tasks, and privacy protection.

Resemblance

To evaluate the resemblance of synthetic data generated by a GAN-based model, various methods can be employed. One common approach is to compare the univariate statistical characteristics of the synthetic data, such as mean, median, and standard deviation, with those of the real data [38]. Additionally, distance calculations, statistical tests, and visual comparisons can also be utilized to assess the resemblance of the synthetic data to the original data [51].

The most commonly used approaches within the resemblance dimension are dimension-wise (DW) probability and distance metrics [3]. DW testing involves comparing the probability distributions and statistical characteristics of each feature between the synthetic and real data. Studies conducted by Ghosheh et al. [19] and Coutinho-Almeida et al. [3] have indicated that the Bernoulli's success probability and chi-squared (x2) test are frequently used for binary features during DW testing. Additionally, the Student T-test is commonly used for continuous features. For comparing the characteristics of features, Hernandez et al. [51] suggested several statistical tests for both numerical and categorical attributes. Numerical attributes can be compared using tests such as Student's T-Test, Mann-Whitney U-Test, and Kolmogorov-Smirnov

(KS) test for mean, population, and distribution comparison. Categorical features can be compared using x2 tests to assess feature independence between real and synthetic categorical data [51, 15, 67]. Additionally, distance metrics, such as Cosine Distance, can be used for DW testing, with smaller distances indicating better preservation of univariate statistical characteristics [51].

Along with DW testing, it is also important to check the real data distributions in terms of the joint distribution [19]. This means that the relationships and patterns between features in the real data should be preserved when synthesizing data. Some metrics that have been used for this are Jensen-Shannon Divergence (JSD), Inception score (IS), Wasserstein Distance(WD), and Maximum Mean Discrepancies (MMD) [19].

Along with DW-testing and joint distribution preservation, it is also important to preserve the inter-dimensional relationship, as well as correlations between features between the real and synthetic data [19]. Common correlation metrics used are Pearson correlation, Spearman correlation, correlation coefficients, and correlation matrices [51]. The inter-dimensional relationships can also be compared by doing Dimension-Wise prediction tests. This is when an ML model, most commonly a prediction model, is trained on real and fake data, and the model performance is compared. Logistic regression models and decision trees are among the most popular ML models used for this [19].

Just performing analytical methods can be misleading. Hence, it is essential to supplement methods such as plotting and visual comparisons. Distribution plots can be helpful in identifying whether the statistical characteristics of the real and synthetic data are comparable. To determine the similarity between the distributions per column in both real and generated data, one can visualize the cumulative sum of each column [67]. These techniques can be used to examine categorical and numerical features, along with principal component analysis (PCA) and correlation matrices [51, 15].

Machine Learning Utility

Synthetic datasets can be assessed by evaluating their utility in downstream machine-learning tasks and model performance. One way is to compare how well a machine learning model (e.g. logistic regression, decision trees, ANNs) performs on real data versus synthetic data. The model's performance on the synthetic data can then be compared to its performance on the real data [15]. If models trained on synthetic data perform similarly to those trained on authentic data, it suggests that the synthetic data conforms to the underlying data distribution [38]. This kind of machine learning utility testing framework was introduced by Esteban et al. [63] and is called "Train on Synthetic, Test on Real" (TSTR). There is also the reverse method, "Train on Real, Test on Synthetic" (TRTS). However, TSTR is generally considered to be more effective than TRTS because TRTS is less capable of capturing mode collapse. In terms of machine learning utilities, TSTR and TRST are among the most common way to evaluate synthetic datasets [19]. However, Jordon et al. [68] proposed the method of Synthetic Ranking Agreement (SRA) also. This framework evaluates how well a synthetic dataset preserves the ranking of predictive models based on their performance. Alternatively, models can be trained on augmented datasets. Data augmentation involves creating modified or synthetic copies of an existing dataset to increase its size. In this case, the model's performance can be evaluated using the

augmented data, which can be generated using STDG [69].

To evaluate the performance of machine learning (ML) models, researchers commonly use several metrics, such as Area Under the Curve (AUC), F1-score, Area Under the Precision-Recall Curve (AUPRC), Accuracy, and Mean Relative Error (MRE) [3]. Typically, these metrics are applied to evaluate the performance of machine learning models in classification and regression tasks, providing valuable insights into their effectiveness.

Privacy

Privacy evaluation methods examine how similar synthetic data is to real data and assess the risk of re-identification. Researchers evaluate the level of privacy protection that the synthetic dataset offers compared to the real data. The goal is to ensure that the synthetic dataset is not vulnerable to re-identification attacks and maintains the privacy of individuals in the original dataset, especially for sensitive data such as medical records.

Differential Privacy (DP) stands out as the most frequently employed method for evaluating Privacy across reviewed publications. This is also a commonly used metric when evaluating GANs for EHRs research [19]. DP is a framework that aims to protect the privacy of individuals in data analysis. The idea behind this framework is that the analysis protects individuals' privacy if one can substitute items in the data, without changing the significant results of the analysis [9]. However, many publications solely relied on DP as a privacy evaluation technique, it is advisable to incorporate multiple evaluation methods as DP alone does not guarantee the protection of real patients [3, 9].

Another way that researchers have evaluated privacy is to test the robustness by applying attack methods such as membership inference attacks, attribute disclosure attacks, and model inversion attacks [9]. A membership inference attack is an attempt by an attacker to determine whether or not a specific data point was used to train the victim model [70]. An attribute disclosure attack is when an adversary manages to find a link between a sensitive value and a victim [71]. Another type of attack used to test GANs is the model inversion attack, where an attacker repeatedly queries the victim model with different inputs and observes the corresponding outputs. Based on this they can reconstruct the sensitive features of the training data. However, this attack method is less commonly used for GANs due to the challenges posed by the non-convex optimization problem and the high-dimensional nature of the data space, which can make it difficult to achieve accurate and meaningful feature reconstructions [72]. All of the attacks mentioned have been tested in various scenarios, with the most common ones being in either a black box or white box setting [19]. In addition, other techniques for privacy evaluation such as measuring Euclidean distance, checking for exact matches, and employing Nearest Neighbours (KNN) are also commonly referenced in research [3, 19].

Key Considerations When Evaluating Synthetic Datasets

Coutinho-Almeida et al. [3] also examined the number of publications that included health professionals for clinical evaluation of synthetic patients. This was done to assess the practicality of using synthetic patients in a clinical setting. However, their review revealed that only 2 out of 22 publications involved health professionals for

Evaluation Dimension	Category	Method
Resemblance	Dimension-wise testing	Bernoulli x2 test Student T-test Mann-Whitney U-Test KS Test Cumulative distribution Cosine Distance
	Joint-distribution similarity	JS-Divergence Inception Score Wasserstein Distance Maximum Mean Discrepancies
	Inter-dimensional relationship similarity	Pearson Correlation Spearman Correlation Correlation coefficients Correlation matrices Dimension-Wise prediction tests
Machine Learning Utilities	Frameworks	TSTR TRTS SRA
	Classification	F1 Recall Precision Accuracy
	Regression	AUC AUPRC MRE
Privacy	Attacks	Membership Inference Attribute disclosure Model inversion
	Other methods	Euclidean Distance KNN Exact Matches

Table 2.2: Overview of evaluation metrics mentioned in published work.

this purpose. Involving clinicians in the evaluation of datasets is crucial for ensuring the reliability of the generated datasets. Therefore, it is important to have clinicians actively engaged in the evaluation process.

Moreover, Hernandez et al. [15]'s review revealed that performance, in terms of footprint and computational cost, is not typically assessed in the context of STDG. Additionally, privacy emerged as the least explored and utilized evaluation criterion across all reviewed publications, whereas utility and ML evaluation was the most commonly employed. Taking all of this into account, it is evident that generating high-quality tabular datasets could be possible. However, incorporating privacy considerations significantly complicates this task. It is important to establish a more optimal balance between privacy and data similarity to mitigate the risk of information leakage through potential adversarial attacks [19].

Also, currently, there are no established benchmarks or standardized metrics for evaluating and comparing the various approaches for resemblance, utility, and privacy. In medical research, there are challenges as researchers often rely on GAN metrics designed for tasks such as imaging or non-medical time-series analysis. These metrics may not be well-suited for evaluating medical data, and researchers may even introduce their own metrics, leading to potential inconsistencies and limitations in evaluating the performance of GAN-generated data in the medical domain [19].

2.5 Synthetic Data in Healthcare

The core focus of this thesis is to investigate the generation of synthetic data in the context of healthcare research. The utilization of synthetic data in healthcare presents complex challenges that do not have straightforward solutions. While generative models can potentially address the issue of data scarcity by generating synthetic data, there are numerous constraints due to regulatory requirements that hinder the creation of larger and more diverse datasets [6]. Morevoer, issues such as ethical considerations, including patient privacy and potential biases, need to be carefully addressed.

When it comes to data sampling, there is a need for thorough scrutiny of the data that is being used to train models. This is because sample-selection biases can arise, due to there being a lack of representative data of the entire population [5]. For example, a serious bias that can happen is that a model used in a hospital starts to admit certain socioeconomic backgrounds. A model could also just be trained with images from a particular piece of equipment, and the model will then have a bias in favor of that particular equipment. This can make for a not generalizable model. Models that have these kinds of biases are not suitable to use with data that has not been used in the training process, as the results will not be fair and correct.

As mentioned previously in Section 2.4, biases can also arise from class imbalances. AI models that are supposed to be used for prognosis or diagnosis, can then have trouble when it comes to rare diseases [5]. This underscores a crucial point: the performance of AI models is only as effective as the quality of the training data they have been provided with.

The approval of AI algorithms in medical devices has been on the rise, and synthetic data can be utilized to enhance the performance of these algorithms. The notion

that current GAN models are capable of accurately capturing data distributions while ensuring high patient privacy is not yet a reality. In fact, there are concerns that such models may even introduce vulnerabilities, such as the risk of patient re-identification [5]. One thing that can happen is a membership inference attack. This attack is a type of privacy breach where malicious actors can utilize publicly available weights of a medical model to synthesize real and private health information. This synthesized information can then be used to disclose sensitive data, resulting in a serious privacy violation [73].

In a 2021 article by Chen et al. [5], challenges in adopting synthetic data in healthcare were highlighted. The authors discussed how synthetic data may be used as a temporary solution for fine-tuning models until better data or alternative measures are available. However, a concern raised is that the quality of the synthetic data may not be adequately evaluated in terms of professional clinical standards. There may not even be any clinical reference standards when it comes to data for new or very rare diseases. Evaluation metrics used for evaluating GAN models are not straightforward to interpret by clinicians. Because of all this, the synthetic data may not reflect specific failure modes. To gain acceptance for synthetic data in healthcare, evaluation tests and metrics should be designed for easy understanding by clinicians, helping to build trust in the use of such data [19].

The article also discusses the use of visual Turing tests as a means of evaluating synthetic data, particularly in the context of image data. However, this process would be extremely tedious if the dataset consists of thousands of images. Additionally, assessing image data may not be straightforward. In contrast, employing Turing tests for tabular health records can pose even greater challenges due to the less intuitive nature of tabular data compared to images [19].

To prevent biases and improve model generalization, it could help to collect data from many different healthcare institutions and organizations, instead of just collecting it from a single one [19]. An issue here is the problem with sharing data between different institutions, as this would go against many data protection laws [5].

This section has mostly talked about using synthetic data as a way to augment real data to fine-tune models, as well as privacy concerns. Today, these are the main ways generative models have been employed in healthcare. However, synthetic data has the potential for various future use-cases. It could be effectively utilized for stress-testing AI algorithms, simulating diverse scenarios in virtual environments, and training AI models to learn from surgical errors without endangering patients [5].

Chapter 3

Approach

This chapter introduces the generative STDG methods employed in this thesis, as well as the evaluation metrics utilized to assess the quality of the generated data. Additionally, a custom evaluation framework, "SynthEval: Synthetic Data Classifier Evaluation", specifically developed for this thesis, will be detailed. Comprehensive implementation information for the chosen methods, evaluation metrics, and the SynthEval framework is presented throughout this chapter.

Python version 3.8 was chosen as the primary programming language for this thesis, due to its compatibility with the required libraries and tools used throughout the research. The implementation details, results, and datasets, including the SynthEval framework, are all available on the thesis' repository[1].

3.1 Experimental Datasets

For this thesis, datasets related to healthcare have been chosen to explore the effectiveness of different STDG models. Working with diverse datasets of varying sizes and complexities can provide a comprehensive understanding of the strengths and weaknesses of the chosen models. By starting with a smaller and simpler dataset and gradually progressing to larger and more complex ones, it will be possible to systematically explore different data and gain insights into the performance of the chosen models in different contexts.

Three datasets of different sizes, feature complexities, and quality were selected as the experimental datasets. Figure 3.1 provides an overview of the class distribution for each dataset.

Lower Back Pain Symptoms Dataset
The initial dataset used for experimentation was the *Lower Back Pain Symptoms Dataset* [2], which is the smallest dataset, comprising only 310 observations and 13 features. All the features are numerical except for one which is the binary class label used for prediction. The dataset comprises data related to potential causes of lower back pain, commonly known as lumbago. The causes of lumbago are multifaceted, ranging from ligaments, muscles, nerves, nearby organs, bones, discs, to tendons in the lumbar

[1] https://github.com/mareped/STDG
[2] https://www.kaggle.com/datasets/sammy123/lower-back-pain-symptoms-dataset

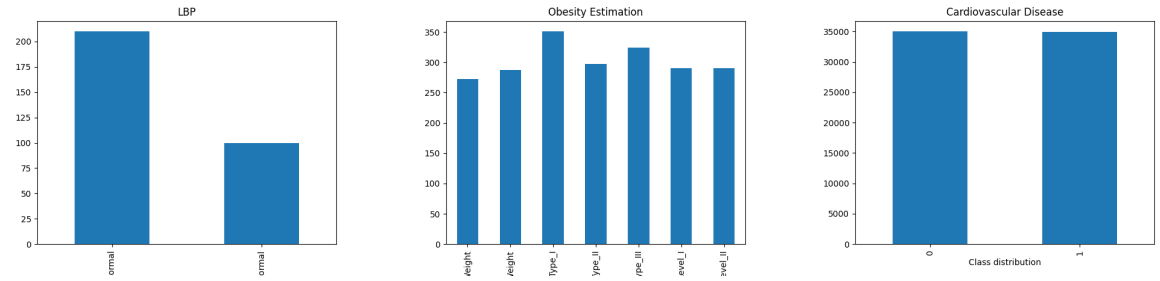


Figure 3.1: Target class distributions of the datasets. *Lower Back Pain Symptoms* (left), *Obesity Prediction* (middle), *Cardiovascular disease prediction* (right).

spine. The data collected in this dataset is intended for predicting the presence of abnormal physical spine data that may contribute to the occurrence of lumbago. Considering the relatively small size of the dataset, it serves as an ideal testing ground for one of the central problem statements of this thesis: enhancing performance through synthetic data augmentation. The dataset's inherent class imbalance, another common characteristic in healthcare data, also enhances its applicability for this study.

Estimation of Obesity Levels

The second dataset used was *Estimation of obesity levels based on eating habits and physical condition* [74]. There are 2111 records in the dataset, with a total of 17 features. These include 8 numerical and 9 categorical attributes, all related to factors that can potentially lead to obesity. Each record is labeled with the class variable NObesity, representing the different levels of obesity a person can be classified as. The dataset is therefore used for multi-class classification problems with the different levels being: Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III. Despite its relatively modest size, it is larger than *Lower Back Pain* and can potentially lead to improved performance in classification tasks. The dataset also provides a valuable opportunity to evaluate the performance of STDG in generating synthetic data for diverse feature types, as well as in more complex classification scenarios.

Cardiovascular Disease Prediction

The last dataset used is the *Cardiovascular Disease dataset*, which can be found on Kaggle[3]. This is a dataset that is used to predict cardiovascular disease. It is also the largest dataset utilized for experimentation, encompassing 70,000 records. The dataset is compromised of 11 features, out of which 5 are numerical and 6 are categorical. The input features encompass both objective information, such as factual data and medical examination results, as well as subjective information provided by the patients. This dataset serves primarily as a benchmark for comparison against two other datasets. Being a relatively large dataset, it is expected to yield better results compared to smaller datasets in theory. However, it is crucial to include a comparative dataset to assess if STDG is effective not only with large datasets, but also with smaller ones. This allows for a comprehensive evaluation of STDG's performance across different dataset sizes.

[3]https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

Dataset name	Records	Numerical	Categorical	Total Features
Lower Back Pain	310	12	1	13
Obesity Estimation	2111	8	9	17
Cardiovascular disease	70 000	5	6	11

Table 3.1: Overview of experimental datasets, with counts of numerical and categorical features.

3.2 Model Selection

The selection of STDG models for this thesis was based on several criteria, including versatility in handling different data types, recentness and support from previous research, and the availability of implementation code. Although various tabular GANs were discussed in Chapter 2.4, not all had functional or easily usable code. Thus, CTGAN and CopulaGAN were chosen because they met the outlined criteria and had straightforward implementation. In this section, a detailed explanation of how these models work will be provided.

3.2.1 CTGAN

The CTGAN model, proposed by Xu et al. [60], is an improved version of TGAN. Synthetic data generated by TGAN often led to mode-collapse and did not have enough variety in the generated data. This was typically an issue when it came to discrete variables. CTGAN was developed by the same authors as TGAN to address the shortcomings of the latter and tackle other challenges involved in using GANs for tabular data generation. Challenges addressed were including mixed data types, modeling non-Gaussian distributions, capturing multimodal distributions, effectively learning from sparse one-hot-encoded vectors, handling highly imbalanced categorical columns, and addressing the lack of ground truth for evaluation.

Tabular data often follows non-Gaussian distributions, which poses a challenge for modeling such data. Continuous tabular data is more complex than image data, as it does not conform to the simpler Gaussian distribution often observed in images. Therefore, applying the same techniques used in GANs for images, such as adding a tanh function in the output layer and performing min-max transformation, to tabular data can lead to the issue of vanishing gradients. Additionally, Capturing multimodal distributions is also an issue with GANs. To handle difficult distributions, the creators of CTGAN invented a mode-specific normalization technique. Mode-specific normalization works in three steps, as explained by Xu et al. [60]:

1. Continuous columns use a Gaussian mixture model (VGM) to estimate the number of modes to fit the resulting mixture.
2. For each data point in the continuous column, the probability of it coming from each mode is computed, based on the probability densities from VGM.
3. One mode is sampled from the given probability density and this mode is used to normalize the value.

The other issue with GANs that they addressed was that categorical data often is represented as one-hot encoded vectors, which can result in sparse and high-

dimensional data. There is also the issue of a lot of categorical data being imbalanced. To address these problems, CTGAN uses a combination of techniques. The approach involves utilizing a conditional generator, along with a training-by-sampling technique.

The conditional generator is used to ensure an even distribution of categories in the data during training. This is done by utilizing a conditional vector, which enables conditioning on a specific value of a particular column through one-hot encoding. The information from the vector is then fed into the generator.

The training process of the generator in CTGAN involves a method known as "training-by-sampling". It is an iterative process where real data distribution samples are drawn, and the generator is informed about the desired category for the generated samples through a conditional vector. The weights of the generator are then updated accordingly. As the training progresses, the generator learns to generate samples that align with the specified categories from the discrete attribute. Additionally, the model is trained using the WGAN loss with Gradient Penalty.

3.2.2 CopulaGAN

CopulaGAN is a version of CTGAN that utilizes Cumulative Distribution Function (CDF)-based transformation, applied via GaussianCopula [61]. CDF is a function that gives the probability that a random variable takes on a value less than or equal to a specified value [75]. On the other hand, a copula is a function that joins multivariate distribution functions to their one-dimensional marginal distributions, simplifying the modeling of the dependence structure between variables [76].

By utilizing the GaussianCopula to transform the data, CopulaGAN is able to learn the data distribution more effectively than CTGAN. This is because CopulaGAN captures the relationships between variables in a transformed space that follows a multivariate Gaussian distribution, making it easier to learn and model the dependencies between the variables.

3.3 Implementing the GAN Models

The SDV library [61] was used to implement, configure, and train both CTGAN and CopulaGAN to generate synthetic datasets. The pre-processing steps and training procedures used for these models are discussed in detail in this section.

3.3.1 Pre-processing

Prior to model training, the data underwent pre-processing. Although all datasets used in this thesis were obtained from open-source websites and have undergone initial pre-processing, such as the removal of null values, additional data preparation steps were also utilized. During the model fitting, the SDV library incorporates some pre-processing steps. For the library to identify categorical and numerical columns, a metadata file must be generated to map the data accordingly. Based on the mapping, categorical columns were encoded with the Scikit-learn LabelEncoder, while numerical columns were normalized using Min-max scaling and transformed into a vector format suitable for model input. These pre-processing steps enhance model performance and accuracy by reducing noise and rendering the data more suitable for the model to learn from. In addition to this, CopulaGAN implements additional

pre-processing steps which involve utilizing GaussianCopula to transform the data, as explained previously.

3.3.2 Training

The two models, CTGAN and CopulaGAN, were initially trained using all three experimental datasets: *Lower Back Pain*, *Obesity*, and *Cardiovascular disease*. Once the training process was complete, the models were saved along with their respective loss values. Subsequently, these saved models were loaded to generate synthetic data that matched the size of the original datasets.

Figure 3.2: Stable (top) vs Unstable GAN (bottom) [1].

As grid search was not compatible with the models from the SDV library, the parameter tuning of the GAN models was executed with a more manual parameter search. To determine the optimal combination of epochs and batch sizes for each model, they were trained with different parameters. Following the training process, a loss vs epochs graph was generated, allowing for observation of the point at which model performance stabilized and where epochs should be cut off. Since the models involved are GANs, both the loss of the generator and discriminator were plotted against each other. After running through enough epochs, the loss vs epochs plot for both the generator and discriminator should eventually stabilize. The first time when training, both models were trained on a large number of epochs, so that one could look at the plot and see where the model starts to stabilize. Typically, the generator begins with a positive loss, but it should decrease with each epoch until it eventually stabilizes at a negative loss. On the other hand, since the discriminator and generator are adversaries, the discriminator's loss should remain stabilized at around zero. This represents an ideal and stable GAN, indicating that as the generator improves, it becomes increasingly challenging for the discriminator to distinguish between real and synthetic data.

A comparison between a stabilized and non-stabilized GAN was presented on SDV's Github discussion page [1], where the losses were plotted. Figure 3.2 illustrates this comparison, where the red line represents the discriminator's loss and the blue line represents the generator's loss. The figure reveals that the non-stabilized GAN's loss becomes increasingly noisy over time, implying that the model is struggling to capture the underlying patterns in the real data.

3.4 Evaluation Framework

The evaluation was approached through three distinct dimensions: Resemblance, Classification, and Privacy. These components test the resemblance of the real and synthetic data, compare the data's performance on various classification models, as well as evaluate the privacy of the synthetic data. The inspiration for the different metrics utilized within the different dimensions was taken from the evaluation metrics discussed in Section 2.4.3.

The metrics were implemented by utilizing available libraries online such as SDMetric [77], which is an evaluation library created by the same researchers behind the SDV library, and the TableEvaluator library [78]. The SDMetric library and TableEvaluator were useful in covering many of the fundamental evaluation metrics, particularly in the resemblance dimension. However, in order to address evaluation metrics and dimensions that these libraries did not cover, certain metrics and a custom framework were developed as well. Below is an outline of the evaluation plan for the synthetic data and its components. The subsequent sections will delve into each step in detail, providing a thorough explanation of the evaluation approach.

- **Resemblance**: This component includes basic statistical checks and analysis of feature distributions to assess the similarity between the real and synthetic data.
- **Classifier Evaluation**: This component evaluates the performance of various machine learning classifiers on the synthetic data using TSTR/TRTS evaluation and ML classification models.
- **Privacy**: This component assesses the privacy risk of the synthetic data by comparing exact matches and nearest neighbors to the real data.

3.4.1 Resemblance

Basic Statistical Checks

First, basic statistical metrics were compared between each dataset and the synthetic data generated by the models. One such metric was visualizing the column-wise means and standard deviations on a log-scale. If the data points follow the diagonal line in the plot, it indicates that the real and synthetic data follow the same underlying univariate statistical characteristics. While this evaluation metric may be considered basic, it can provide valuable insights into the overall performance of the model. Poor results from this metric may indicate potential issues with other evaluation metrics as well.

Another initial check was looking at the correlations between the real and synthetic data. Here the Pearson correlation coefficient was used. Column correlation distances are used to measure the differences in the column correlations between the data. Specifically, the metrics used are Root Mean Squared Error (RMSE) and Mean Average

Error (MAE). Lower values of RMSE and MAE would indicate that there are small differences between the data, meaning that the synthetic data is relatively similar to the real data.

The correlations were visualized through a correlation matrix where each column is compared against each other, as well as its corresponding difference matrix. The difference matrix was obtained by subtracting the synthetic correlations from the real ones. When analyzing the difference matrix, strong positive values indicate a stronger correlation in the real dataset compared to the synthetic dataset, while strong negative values indicate the opposite. An ideal difference matrix would have values close to zero, which indicates a similar level of correlation between the real and synthetic datasets. Additionally, a scatter plot for all the correlation coefficients was also plotted. If the data points follow the diagonal line, it means that the correlations between variables are similar between the data. However, this measure is more informative about the data's ability to capture the underlying structure of the correlations, not the relationship between columns.

Feature Distributions
A way to see how well the synthetic data manages to capture the statistical properties of the real data is to visualize the distributions of each feature and compare them to the synthetic data. For this thesis, the real and synthetic data distributions and cumulative sums of each feature were plotted on top of each other. If the distributions and cumulative sums of the real and synthetic data overlap, it suggests that the synthetic data provides an accurate representation of the real data.

To quantify the differences in the distributions of each feature, The KSComplement and TVComplement were also used. The Kolmogorov-Smirnov test is applied by KSComplement on numerical data and measures the distance between the two cumulative distribution functions. The TVComplement is used for categorical data and calculates the Total Variation distance between the two probability distributions. Higher values indicate a more similar distribution.

3.4.2 Classifier Evaluation

The performance of synthetic data was also examined in the context of machine learning prediction tasks by comparing it with the performance of the real data. If the synthetic data performs similarly, it implies that the synthetic data has effectively captured the real data's underlying statistical properties and distributions. The TableEvaluator library provides insights into classifier performance, where both real and synthetic data are used for training and testing. However, the classifiers used in this evaluation are pre-set, and their parameters and settings cannot be adjusted to optimize performance. In addition, the library only shows F1 scores and does not allow for the inclusion of other metrics.

In order to offer increased flexibility in the choice of classifiers and metrics used, a custom evaluation framework was developed for this thesis, referred to as "SynthEval: Synthetic Data Classifier Evaluation". This framework is designed to train and test classifiers on both real and synthetic datasets and has been extended to evaluate the potential benefits of augmenting real datasets with synthetic data. The framework is user-friendly, allowing users to easily specify the desired number of classifiers and their respective configurations. There is no imposed limit on the number of classifiers

that can be utilized. Additionally, the paths to the real and synthetic data must be provided.

To configure the evaluation process, one can determine the proportion of real and synthetic data to be used for augmentation. In the example provided below, 50% of the real data and 100% of the synthetic data are employed for the augmented dataset. The *cross_val* parameter enables one to utilize cross-validation if set to *True*. By default, the framework trains each classifier using a traditional Train/test split. After completed code execution, all the results from the evaluation in the form of F1 scores and ROC curves will be saved in a folder. The following sections will delve into a more detailed explanation of the underlying processes within the framework.

```
logreg = LogisticRegression()
rf = RandomForestClassifier()
mlp = MLPClassifier()

evaluator = SynthEval(real_path, synthetic_path, result_path)

evaluator.add_all_classifiers(logreg, rf, mlp)

evaluator.compare_datasets_performance(real_percentage=0.5,
    synth_percentage=1, cross_val=True)
```

Listing 3.1: Code snippet demonstrating the utilization of the custom framework "SynthEval" for evaluating synthetic data.

Training and Testing Phases

The proposed framework streamlines the training and evaluation process by dividing it into three well-structured and systematic phases. Each phase focuses on training a classifier on real, synthetic, and augmented data. This framework builds upon the concepts of "Training on Synthetic, Testing on Real" (TSTR) and "Training on Real, Testing on Synthetic" (TRTS), as explained in the previous Section 2.4.3. Alongside these concepts, the framework also incorporates training on augmented data and testing on real data to explore whether enhancing real data with synthetic data can further improve the classifier's performance. This comprehensive approach allows for a comparison of the classifiers' effectiveness and adaptability across real, synthetic, and augmented data.

Phase 1: Training and Evaluation on Real Data In this phase, the classifier is trained using the real dataset, which is divided into a training set and a test set according to standard evaluation practices. The classifier's performance is assessed on both the real dataset's test set and an additional synthetic test set, following the TRTS concept.

Phase 2: Training and Evaluation on Synthetic Data During this stage, the classifier is trained with the synthetic dataset, which is also partitioned into a training set and a test set. The model's performance is evaluated on the synthetic dataset's test set and the real dataset, in line with the TSTR concept.

Phase 3: Training and Evaluation on Augmented Data In this phase, the classifier is trained using the augmented dataset, which combines real and synthetic data. The augmented data is created based on the desired proportions of real and synthetic data. During the data creation process, care is taken to ensure that none of the real test data is present in the augmented dataset. This approach avoids data leakage and potential

overfitting during the evaluation process. The remaining real data and the augmented data are effectively separated, keeping the real data used for testing distinct from the real data used in the augmented dataset. The classifier's performance is then assessed on the test sets from both the augmented and remaining real dataset.

Each of the three phases is repeated for every classifier assigned to the framework, ensuring a thorough evaluation of all classifiers across all the different data.

Pre-processing and Evaluation Methods
Before training each dataset, the framework preprocesses the data using Sci-kit's LabelEncoder and MinMaxScaler. As illustrated in the code example above, users have the option to choose between Sci-kit's Train/Test split or k-fold cross-validation for evaluating the classifier during each training phase. Train/Test split is a faster method that provides a quick estimate of the model's performance. As described in each phase, the training set is composed of the data being trained during that specific phase in the framework. Meanwhile, the test set encompasses the extracted test set, as well as an additional test set for comparison.

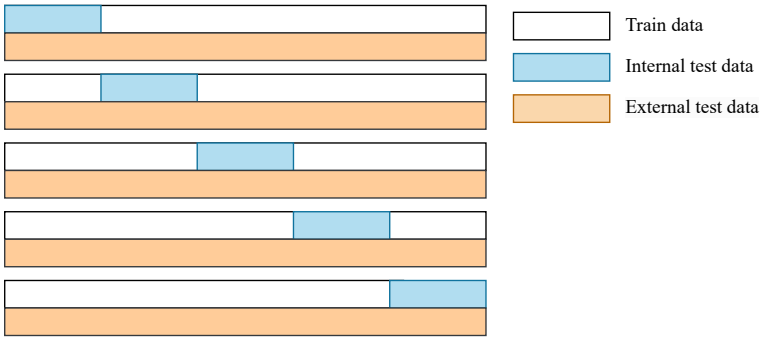


Figure 3.3: Cross validation within the SynthEval framework.

On the other hand, cross-validation offers a more robust evaluation method, as it divides the data into k-folds, training the classifier on k-1 folds and evaluating it on the remaining fold. This process is repeated for a specified number of folds, typically 5 or 10. While cross-validation is more computationally demanding, it yields a more accurate assessment of the classifier with reduced bias compared to the Train/Test split method. This is especially beneficial for small datasets, as it optimizes the use of available data. Figure 3.3 illustrates the cross-validation process within the SynthEval framework. The cross-validation process consists of the following steps:

1. Train the classifier on k-1 folds of the data.
2. Test the classifier on the remaining fold of the data being trained (internal test data).
3. Evaluate the classifier on the supplementary test data provided within the same fold (external test data).
4. Repeat steps 1-3 for each fold in the cross-validation process. After all folds have been processed, average the predictions for the external test data across all folds to produce a final set of predictions.

This approach ensures an in-depth evaluation of the classifier's performance throughout each phase of the framework, leveraging the robustness and adaptability of cross-validation. The framework also utilizes Stratified k-fold cross-validation, which strives to preserve a similar proportion of samples from all target classes in each set, making it well-suited for imbalanced datasets.

Evaluation Results

The evaluation results are presented in the form of F1 scores and ROC curves, which are visualized for easy comparison and analysis. The F1 score is chosen because it is a metric that provides a balanced average of precision and recall. The difference in F1 scores between the real data and the synthetic/augmented data is also computed. This gives an indication of how similar the dataset performs. If the F1 difference is positive, then the classifier performs better on the real data. If the difference is negative, the classifier performs better on the synthetic/augmented data. If the value is close to zero, the performance is similar on both datasets.

ROC curves, short for Receiver Operating Characteristic curves, offer a visual representation of classifier performance. ROC curves are particularly useful in healthcare and diagnostic systems because they provide a comprehensive view of the trade-off between sensitivity (true positive rate) and specificity (false positive rate) at different thresholds [79]. This is crucial as false positives and false negatives can have severe consequences in medical settings. Additionally, ROC analysis is not influenced by class distribution, making it well-suited for healthcare data, which is often imbalanced.

ROC curves are intended to be used in binary classification problems, however, it is possible to use them for multi-class problems as well by utilizing a one-vs-all (OvA) strategy [80]. Each class can be evaluated by generating a separate ROC curve that treats that class as the positive class and the other classes as negative classes, essentially treating the problem as multiple binary classification problems. The framework allows the user to generate a curve for each class, but visualizing many curves can become messy and unorganized when dealing with numerous classes. To make the plot more readable, micro and macro averaging of all the target classes are plotted as a default for multi-class classification problems. Micro averaging should be used over macro averaging when the classes are highly unbalanced. This is because micro-averaging is more sensitive to the performance of the minority classes, while macro-averaging is more sensitive to the majority classes.

Selected Classifiers for Evaluation

For this thesis, three different classifiers were chosen to evaluate the different datasets. These were Random Forest, Logistic Regression, and the MLP Classifier from the Sckikit-Learn library [81].

Logistic Regression is a linear statistical model that predicts the probability of an event occurring by fitting the data to a logistic function. This method is commonly employed for binary prediction problems, but can also be extended for multi-class classification problems as well. The main advantages of this model is its low computational costs, simplicity, and interpretability [82]. On the other hand, Random Forest is an ensemble method that enhances prediction accuracy by aggregating multiple decision trees. This more complex model can capture intricate data patterns, potentially

leading to superior performance over logistic regression. Particularly well-suited for large or high-dimensional datasets, Random Forest can handle both regression and classification tasks [83]. Lastly, The MLP (Multi-Layer Perceptron) Classifier is a type of artificial neural network that uses multiple layers to classify data. It can employ various optimization algorithms, such as stochastic gradient descent, lbfgs, or Adam, which were explained in detail in Chapter 2.

By utilizing these three models, the evaluation was done with a diverse set of modeling techniques, each with its own strengths and weaknesses. Logistic Regression provides simplicity and interpretability, Random Forest offers robustness and better handling of complex data, while the MLP Classifier introduces the power of artificial neural networks. This combination ensures a broad evaluation of all the datasets, taking advantage of the unique capabilities of each model.

3.4.3 Privacy

For the privacy dimension, the first two metrics that were looked at were duplicate rows between the real and synthetic data, along with if there are any exact matches between the sets. A dataset with numerous duplicate rows may pose privacy risks, as it can expose underlying patterns or correlations in the actual data, potentially making sensitive information about individuals easier to uncover. If the synthetic dataset does not have many duplicates, it may help mitigate privacy risks. Additionally, by seeing if there are many duplicated rows in the fake data, it can indicate that the generative model has not introduced enough variability in the new dataset, and it may suffer from mode collapse.

Furthermore, looking at exact matches is a way of seeing if the synthetic dataset is copying identical rows from the real dataset into the synthetic one. If the synthetic data is starting to replicate the real data, it would mean that the generative model is not generating different enough data to preserve privacy. This can result in potential data leakage. Also, if there are a lot of identical values between the sets, it may be a sign of overfitting. SDMetrics' NewRowSynthesis is used to measure this, where one can give a numerical match tolerance percentage as a parameter to set how strict the matches should be. This percentage controls how close the numerical values in a row have to be, to become a match. Meaning a numerical match tolerance of 0 would only count values that are exactly the same. The default value is 0.01, which means that if two numerical values have an absolute difference smaller than or equal to 1%, the row is considered an exact match.

The TableEvaluator library provides a metric called the nearest neighbor mean and standard deviation, which was used in this thesis to assess the similarity between synthetic and real data and evaluate privacy. This metric utilizes Euclidean Distance to measure the distance between each synthetic record and the most similar real record. Euclidean distance is a metric often used in related work (see Table 2.2). While the Euclidean distance metric can become less effective with high dimensional data [84], it can still be a useful metric for evaluating the similarity between synthetic and real data. From a synthetic data perspective, the mean and standard deviation of this metric should both be as close to 0 as possible to suggest that the data is highly similar to the real data. However, from a privacy standpoint, a large mean and a low standard deviation is the ideal result [78]. This is because a large mean indicates that the synthetic data is more dissimilar to the real data, and a low standard deviation

indicates that the distances between the data are relatively consistent. This highlights the importance of balancing privacy and data similarity, as discussed in Section 2.4.3.

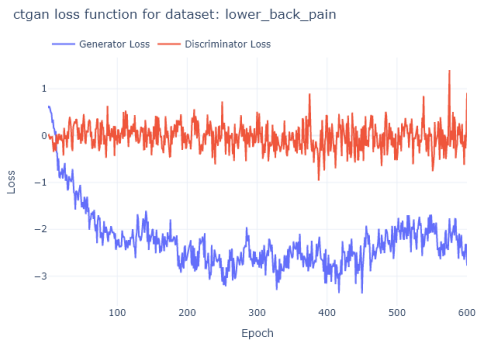
Chapter 4

Results

This chapter presents the results of the training and evaluation process. While all the generated results have been analyzed, only the most significant findings are presented to avoid repetition. The remaining results for all the datasets can be found in Appendix A, B, and C, as well as in the associated GitHub repository for this thesis.

4.1 Training the GAN models

CTGAN



CopulaGAN

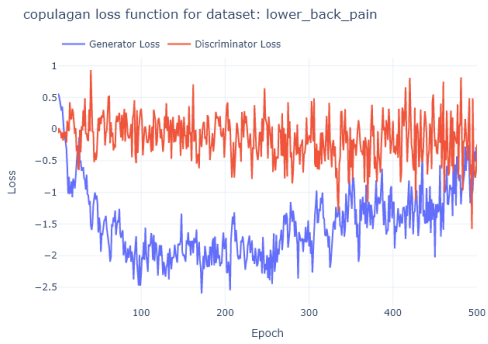


Figure 4.1: Loss curves for the *Lower Back Pain* dataset

On the Lower Back Pain dataset, CTGAN was trained for 600 epochs using a batch size of 100, and CopulaGAN was trained for 500 epochs with the same batch size. Figure 4.1 indicates that CopulaGAN was noisier than CTGAN and did not stabilize. Although the generator and discriminator losses of CTGAN started overlapping around 700-800 epochs, the model was cut off before that occurred.

CTGAN CopulaGAN

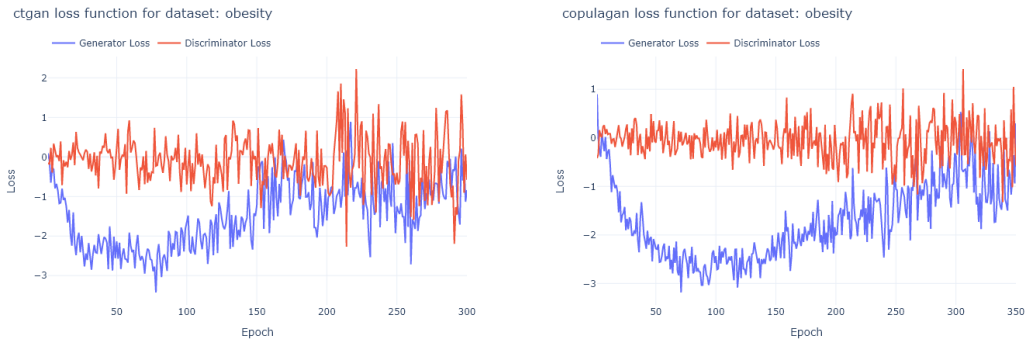


Figure 4.2: Loss curves for the *Obesity* dataset

On the *Obesity* dataset, CTGAN was trained for 300 epochs using a batch size of 50, and CopulaGAN was trained for 350 epochs with a batch size of 100. Both models showed similar training patterns, becoming increasingly noisy around 150-200 epochs.

CTGAN CopulaGAN

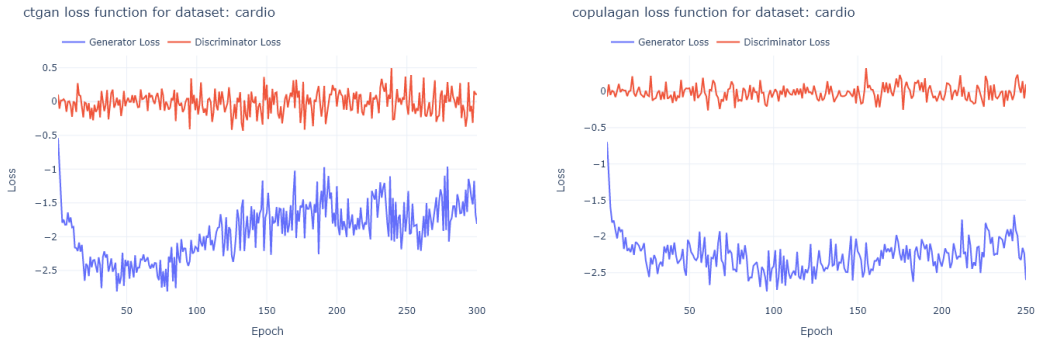


Figure 4.3: Loss curves for the *Cardiovascular Disease* dataset

For the *Cardiovascular Disease* dataset, CTGAN was trained for 300 epochs and CopulaGAN for 250 epochs. Both with a batch size of 400. At approximately 100 epochs, the generator loss of CTGAN started to rise, indicating that the model was not fully stabilized. Although the losses did not overlap, there was an observable trend of instability. In contrast, CopulaGAN's generator and discriminator loss both seem to stabilize during training.

4.2 Basic Statistics

4.2.1 Mean and Standard Deviation

Figure 4.4 displays the means and standard deviations for all datasets generated by CTGAN (refer to Appendix A.2, B.1, and C.1 for CopulaGAN datasets). From the plots, it was observed that both models have captured almost identical means and standard deviations, which is why only CTGAN is shown below. It also appears that both models successfully captured most of these properties with ease, for most

of the generated datasets. However, both models seemed to have some difficulty with capturing the standard deviations of the *Cardiovascular disease* dataset, as there is one outlier observed. The majority of data points align with the diagonal line, but the presence of this outlier suggests potential implications for future results. It raises concerns about the models' ability to effectively represent the true population and the inherent variability in the real data.

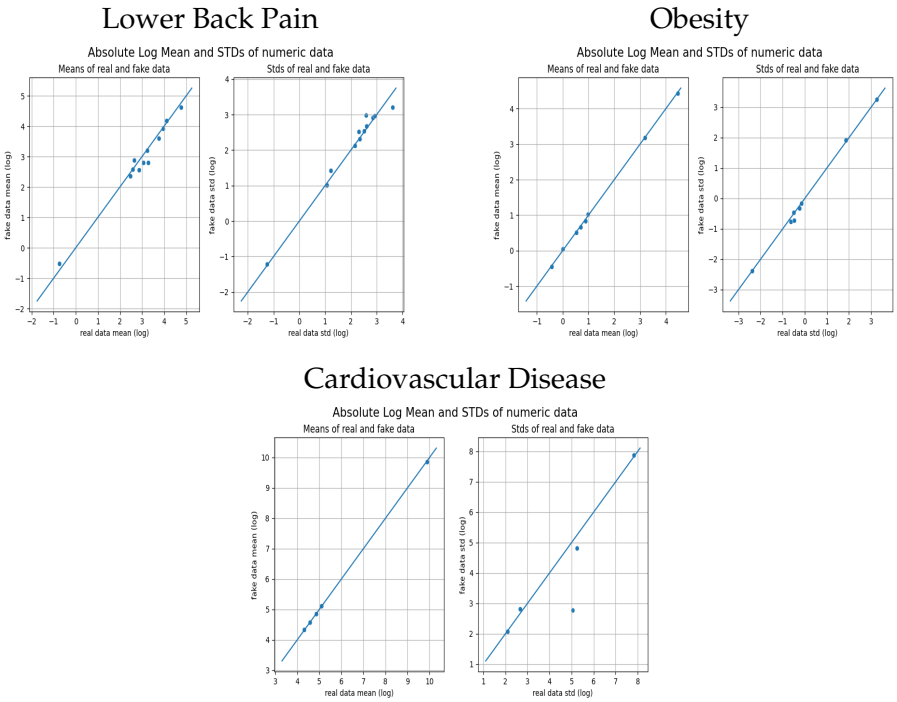


Figure 4.4: Absolute Log Mean and STD of numeric data for datasets generated by CTGAN.

4.2.2 Correlation

Model	**Dataset**	**Correlation Distance**	
		RMSE	**MAE**
CTGAN	Lower Back Pain	0.1980	0.1228
	Obesity	0.0936	0.0625
	Cardiovascular Disease	0.0823	0.0481
CopulaGAN	Lower Back Pain	0.1305	0.0909
	Obesity	0.0918	0.0623
	Cardiovascular Disease	0.0667	0.0321

Table 4.1: Column Correlation Distances.

Table 4.1 shows the correlation distances between the real and synthetic generated data for each model. Overall, the differences between CTGAN and CopulaGAN in correlation distances are relatively small, however, the values for CopulaGAN are

generally lower than those of CTGAN across all three datasets. This indicates that CopulaGAN's synthetic data better approximate the correlation structure of the real data. *Cardiovascular Disease* stands out as the dataset that captures correlations most effectively, evidenced by its lower RMSE and MAE values compared to the other two datasets for both models. However, the *Lower Back Pain* dataset is struggling the most in this regard.

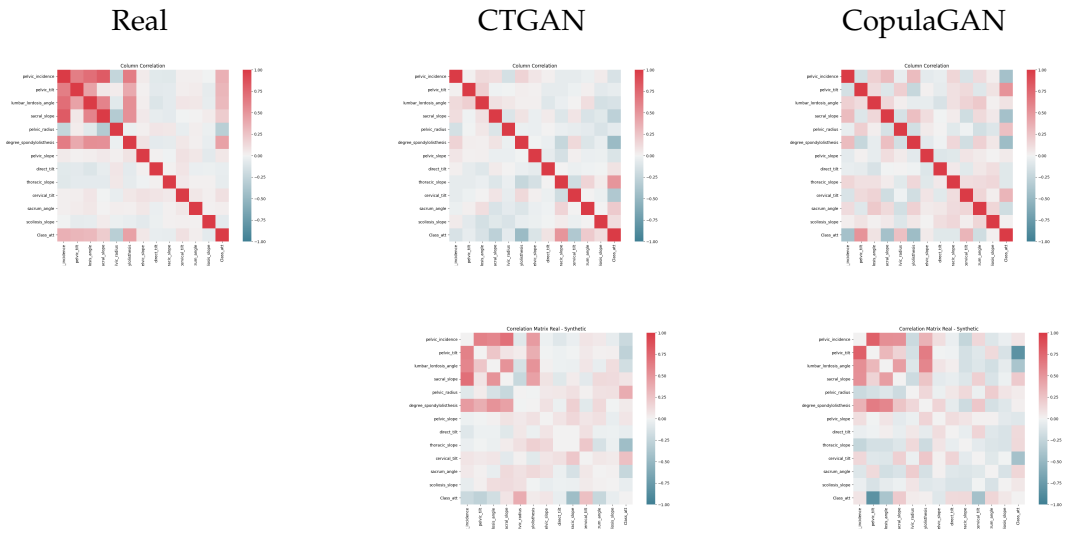


Figure 4.5: Comparison of correlation matrices for the *Lower Back Pain dataset* (top row), along with the corresponding difference matrices (bottom row).

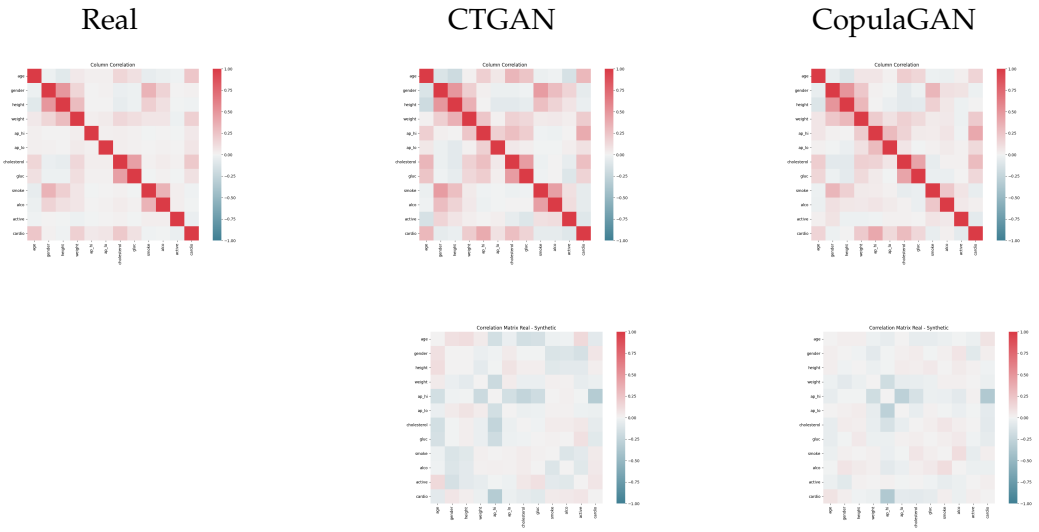
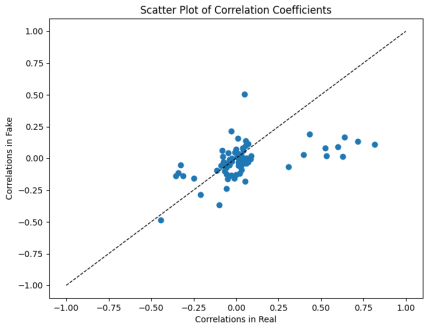


Figure 4.6: Comparison of correlation matrices for the *Cardiovascular Disease* dataset (top row), along with the corresponding difference matrices (bottom row).

Figure 4.5 and 4.6 illustrates the differences in how well the correlations are captured

on both the *Lower Back Pain* and *Cardiovascular Disease* dataset. The visualization of the *Obesity* dataset correlations can be seen in Appendix B.2. These results mirror the RMSE and MAE values, as it is clear that *Lower Back Pain* is struggling. Ideally, the difference matrix should exhibit minimal deviations from zero, resulting in a heatmap with weak color gradients. However, this is not the case for the *Lower Back Pain* dataset. The scatterplot in Figure 4.7, which displays the correlation coefficients between the real and synthetic data, highlights this further. The data points are widely scattered and deviate from the expected diagonal line.

CTGAN



CopulaGAN

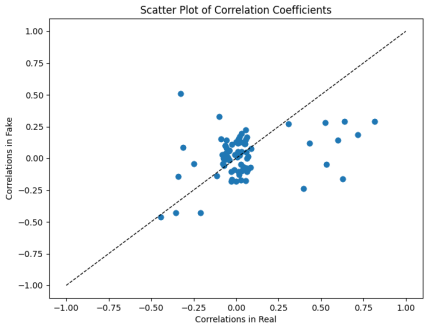


Figure 4.7: Scatter plot for correlation coefficients on the *Lower Back Pain* dataset.

However, the matrices, scatterplots, and RMSE/MAE values for the *Obesity* and *Cardiovascular Disease* dataset shows that both models manage to capture the correlations somewhat. Some differences in correlation are expected, due to the synthetic data being an approximation of the real data, but the results are promising.

4.2.3 Distributions

Model	Dataset	Complement Score
CTGAN	Lower Back Pain	0.785
	Obesity	0.876
	Cardiovascular Disease	0.906
CopulaGAN	Lower Back Pain	0.706
	Obesity	0.869
	Cardiovascular Disease	0.936

Table 4.2: Average of the KSComplement and TVComplement values for all the datasets.

Table 4.2 presents a complement score for each dataset based on the average of the KSComplement and TVComplement values of each column. Both models have comparable scores across the synthetic datasets generated, but CTGAN performs better in capturing the distributions of the *Lower Back Pain* and *Obesity* datasets compared to CopulaGAN. On the other hand, CopulaGAN shows slightly better performance on

the *Cardivascular Disease* dataset. The consistent pattern of *Cardiovascular Disease* outperforming the other dataset, while the *Lower Back Pain* dataset consistently performs worse, is evident. This observation aligns with the expectation that performance tends to correlate with the size and quality of the real dataset.

The distribution and cumulative sum plots were printed for all the columns in each dataset. This section will only show two distinct features from each dataset to visualize the difference in how well CTGAN and CopulaGAN have managed to capture the distributions of the real data.

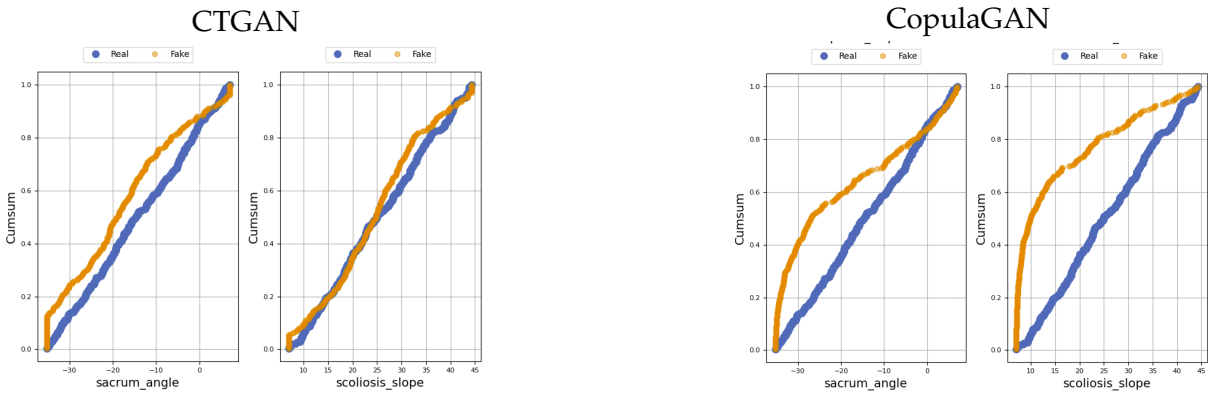


Figure 4.8: Cumulative sums of the features: "sacrum angle" and "scoliosis slope" in the *Lower Back Pain* dataset.

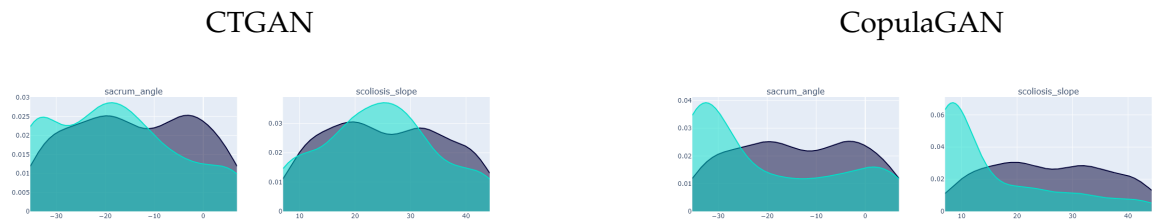


Figure 4.9: Comparison of column shapes for the "sacrum angle" and "scoliosis slope" features in the *Lower Back Pain* dataset.

The charts depicted in Figure 4.8 illustrates the cumulative sums derived from two distinctive features, namely "sacrum angle" and "scoliosis slope", within the *Lower Back Pain* dataset. The same features are shown in the column plots in Figure 4.9. These visualizes the shapes of the columns in the synthetic data against the real.

The plots demonstrate that CTGAN produces cumulative curves closer to each other compared to CopulaGAN. These results indicate that CTGAN is more successful in capturing similar distributions for both features. Although CTGAN seemingly outperforms CopulaGAN in capturing the distributions, the feature curves from the synthetic data do not display smoothness or perfect alignment with the real data curves. Furthermore, the KSComplement values for CTGAN are not consistently high across all features. This implies that although CTGAN performs better, it still falls short of successfully capturing the distributions.

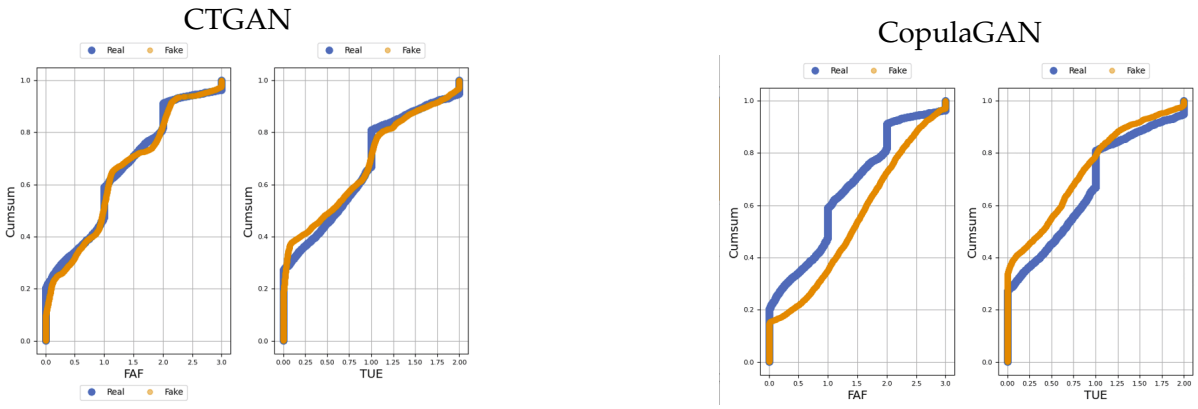


Figure 4.10: Cumulative sums of the features "FAVC" and "TUE" in the *Obesity* dataset.

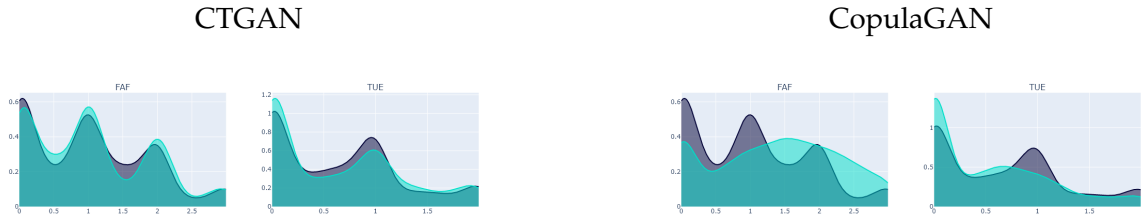


Figure 4.11: Comparison of column shapes for the "FAVC" and "TUE" features in the *Obesity* dataset.

Figure 4.10 and 4.11 shows the cumulative sums and column shapes for the features "FAVC" (Frequency of consumption of vegetables) and "TUE" (Time using technology devices). Again, it is observed that the overlapping is better for CTGAN than CopulaGAN.

Lastly, Figure 4.12 and 4.13 represents the features "Height" and "Weight" for the *Cardiovascular Disease* dataset. The cumulative sum curves and column shape plots for both models are smooth and overlapping. This pattern is observed across all the features in the dataset, indicating that both models are performing well in synthesising the *Cardiovascular Disease* dataset.

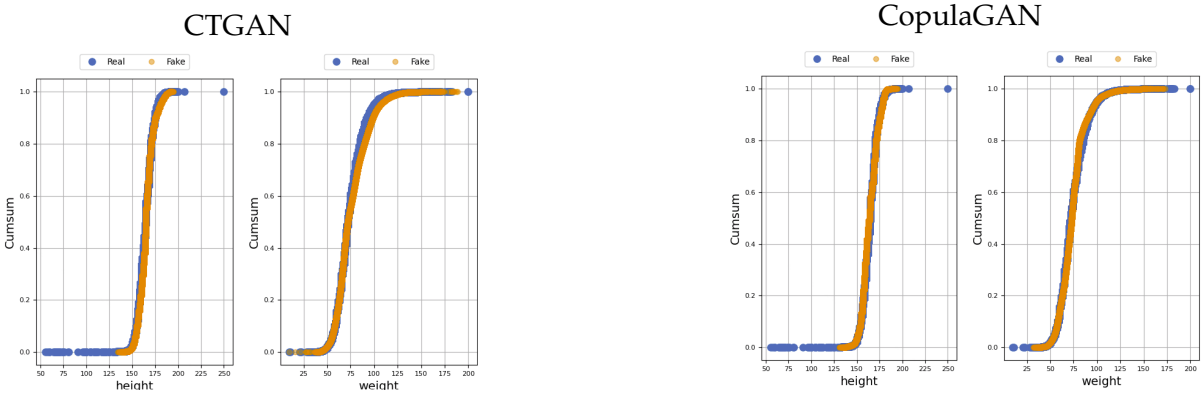


Figure 4.12: Cumulative sums of the features "Height" and "Weight" on the *Cardiovascular Disease* dataset.

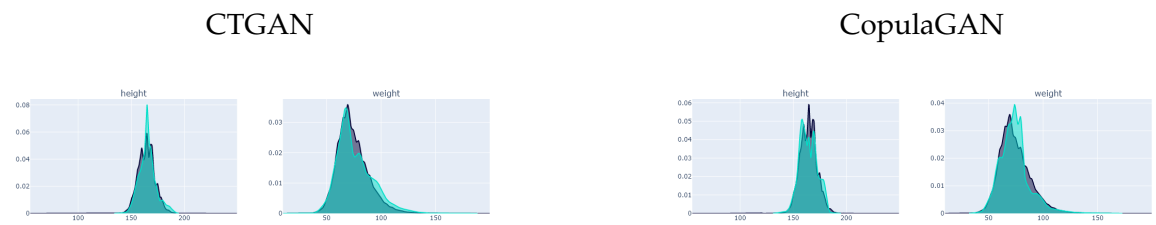


Figure 4.13: Comparison of column shapes for the "Heigh" and "Weight" features in the *Cardiovascular Disease* dataset.

4.3 Evaluating Classifiers on Synthetic Data

This section presents the results obtained from SynthEval, the developed classifier framework for this thesis. The classifiers used to compare the performance of the synthetic dataset are Logistic Regression, Random Forest Classifier, and MLP Classifier. To ensure robust training of each classifier, a cross-validation approach with 5 folds was utilized instead of a traditional train/test split.

4.3.1 Lower Back Pain dataset

All the F1 scores from using the SynthEval framework on the *Lower Back Pain* dataset are summarized in the following table:

		CTGAN		CopulaGAN	
Classifier	**Trained On**	**f1_real**	**f1_synth**	**f1_real**	**f1_synth**
Logreg	Real	0.6866	0.2812	0.6866	0.1922
Logreg	Synthetic	0.5239	0.8904	0.2968	0.8810
RFC	Real	0.7817	0.4188	0.7807	0.2196
RFC	Synthetic	0.5506	0.9201	0.5692	0.9008
MLP	Real	0.7418	0.3589	0.7381	0.2212
MLP	Synthetic	0.5309	0.8955	0.3697	0.8891

Table 4.3: Classifier results for datasets generated by CTGAN and CopulaGAN on the *Lower Back Pain* dataset.

The table reveals that when training on synthetic data and testing on real data, the F1 scores consistently tend to be lower compared to training and testing on real data for all classifiers and GAN models. This trend is supported by the negative differences between the F1 scores for real and synthetic data, indicating that the synthetic data underperforms compared to the real data.

Furthermore, it is also clear that CopulaGAN struggles more than CTGAN, due to the F1 differences being comparatively much higher than CTGAN. For example, the F1 difference for the logistic regression model training on real data has an F1 difference of 0.49, and the MLP classifier with a difference of 0.51. Despite CTGAN outperforming CopulaGAN, the noticeable differences in F1 scores persist for both models. These

findings indicate that neither GAN model has succeeded in generating synthetic data that matches the quality of the real data.

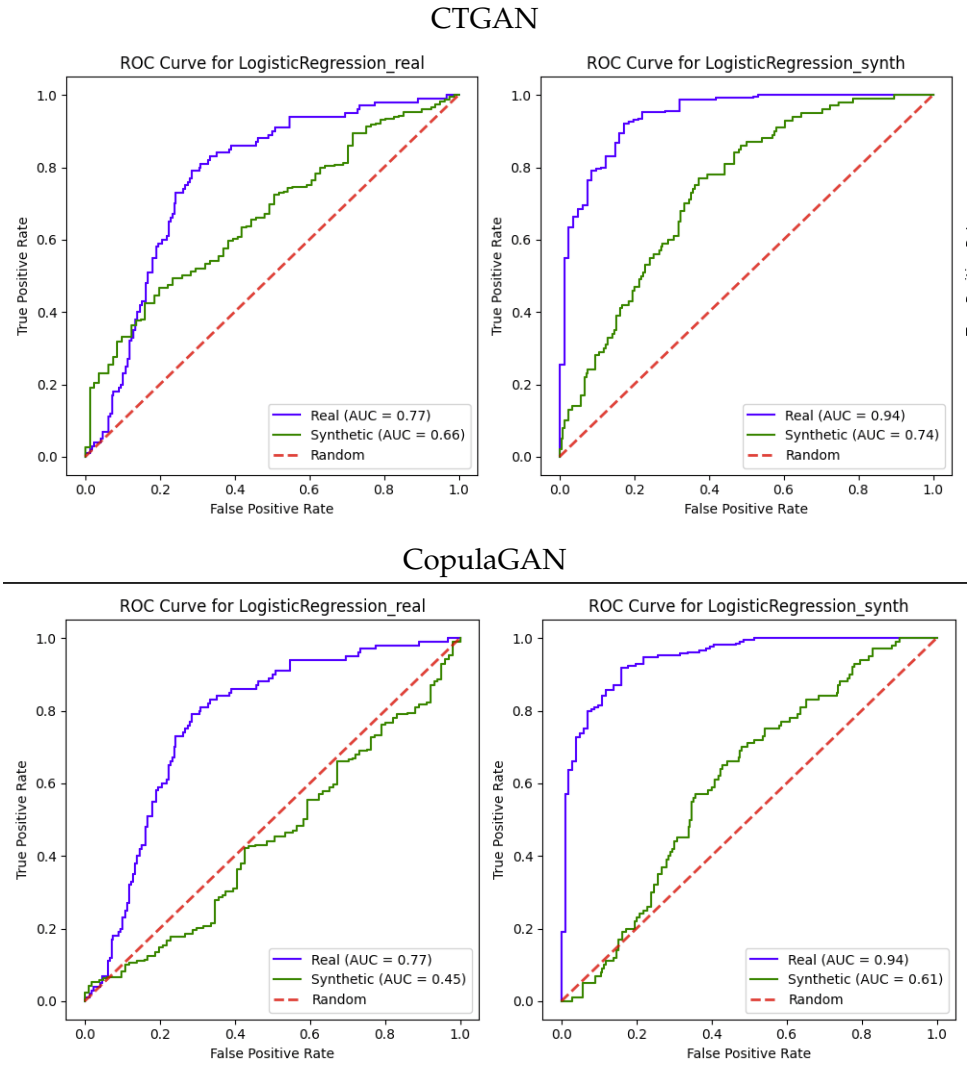


Figure 4.14: ROC plots comparing CTGAN and CopulaGAN datasets on the *Lower Back Pain* dataset. Left: Classifier trained on real data. Right: Classifier trained on synthetic data. Blue curve represents testing on real data, and green curve represents testing on synthetic data.

The ROC curves displayed in Figure 4.14 illustrate a comparison between synthetic data created by CopulaGAN and CTGAN for Logistic regression. It is apparent that CopulaGAN is less effective than CTGAN when examining the curves for testing on synthetic data.

By analyzing the ROC curves, it can be seen that when trained on synthetic data, performance improves when testing on real data compared to training on real data. The AUC value is 0.77 when training and testing on real data, whereas it increases to 0.94 when training on synthetic data. This suggests that the logistic regression model trained on synthetic data generalizes better to real data than the one trained on real data. One possible explanation is that the synthetic data offers a more diverse dataset for the model to learn from. However, the AUC remains low for testing on synthetic

data when trained on both real and synthetic data. This finding suggests that while synthetic data may improve the classifier's performance on real data, it falls short of accurately capturing the intricate complexities. Previous results have revealed that the synthetic data struggles with capturing the correlations and distributions within the *Lower Back Pain* dataset, potentially leading to biases in the generated synthetic data. Possible factors contributing to this issue could include class imbalance, the small amount of data, or the presence of noise within the data.

4.3.2 Obesity dataset

		CTGAN		CopulaGAN	
Classifier	**Trained On**	**f1_real**	**f1_synth**	**f1_real**	**f1_synth**
Logreg	Real	0.6833	0.3481	0.6833	0.3427
Logreg	Synthetic	0.3963	0.3863	0.2968	0.8810
RFC	Real	0.9356	0.3196	0.4376	0.4156
RFC	Synthetic	0.3953	0.4144	0.9378	0.3149
MLP	Real	0.8609	0.3297	0.8696	0.3073
MLP	Synthetic	0.4205	0.3999	0.4451	0.4200

Table 4.4: Classifier results for datasets generated by CTGAN and CopulaGAN on the *Obesity* dataset.

Table 4.4 shows the F1 scores when comparing the synthetic datasets generated based on the *Obesity* dataset. By comparing the results between CTGAN and CopulaGAN, a trend for both models emerges, which indicates that training on real data produces better results than training on synthetic data. This is evident, as the F1 differences are all positive, and in some instances, the differences are significantly high. CTGAN does perform better when training on real data, while CopulaGAN seems to perform better when training on synthetic data in some cases. However, the differences in F1 scores from the GAN models are relatively small, which means that the models seem to perform in a more similar manner compared to the *Lower Back Pain* dataset.

However, it is worth noting that there is a significant exception to these findings. In the case of training on synthetic data generated by CopulaGAN using the RFC classifier and testing on real data, a noteworthy F1 score of 0.9378 was achieved. In contrast, training on real and testing on real data has a score of 0.4737 for the same classifier. Despite this exception, the overall pattern indicates that real data consistently outperforms synthetic data for both CTGAN and CopulaGAN models.

The ROC curves for both CTGAN and CopulaGAN exhibit similar characteristics; therefore, only CTGAN's ROC curves are presented in Figure 4.15 as a representative example of the overall performance. From the plots, it is observed that when training on real data, the micro and macro averages of testing on real data are significantly higher than the macro average when testing on synthetic data. This suggests that the synthetic data is performing well overall, but that there may be specific classes that are unrepresented in the synthetic data. One potential explanation for this phenomenon is that, although the real data might have a balanced distribution of classes, the quality and variety of real data for certain classes could be insufficient. Consequently, the synthetic data might be less effective at replicating these specific classes.

Furthermore, a performance decrease was observed when training on synthetic data and testing the classifier on real data. Despite the AUC values for both synthetic and real testing being comparable, this decline in performance suggests that the synthetic data has not captured all the underlying complexities present in the real data.

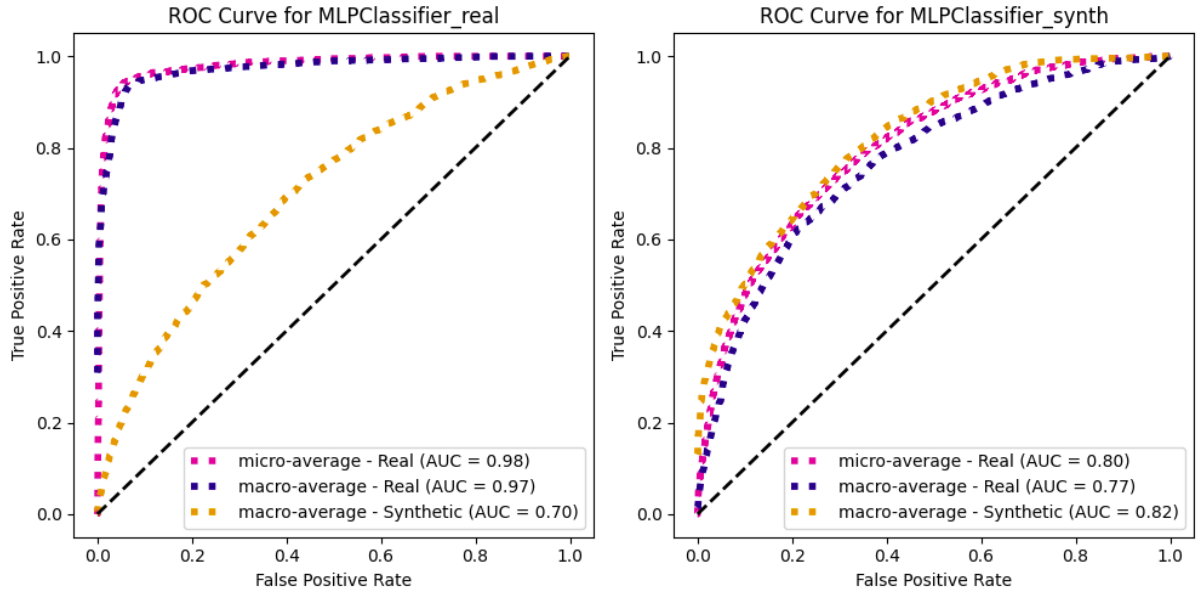


Figure 4.15: ROC plots comparing CTGAN datasets on the *Obesity* dataset. Classifier trained on real data is on the left and trained on synthetic data on the right.

4.3.3 Cardivascular Disease dataset

		CTGAN		CopulaGAN	
Classifier	**Trained On**	**f1_real**	**f1_synth**	**f1_real**	**f1_synth**
Logreg	Real	0.6474	0.6837	0.6474	0.6422
Logreg	Synthetic	0.6957	0.724	0.6893	0.7228
RFC	Real	0.7152	0.7091	0.7155	0.6981
RFC	Synthetic	0.6906	0.7343	0.6872	0.7422
MLP	Real	0.7224	0.7163	0.7215	0.6971
MLP	Synthetic	0.7021	0.7407	0.7048	0.7402

Table 4.5: Classifier results for generated datasets by CTGAN and CopulaGAN for the *Cardivascular Disease* dataset.

Table 4.5 shows the performance of the different classifiers on the synthetic *Cardivascular Disease* datasets. Compared to the other two datasets, the difference in F1 scores between training on real and synthetic data is much smaller for this dataset. Additionally, training on synthetic data and testing on real data results in a performance that is similar to training on real data and testing on real data. In certain cases, there is even a slight improvement in performance. For example, both CTGAN and CopulaGAN-generated data demonstrate enhanced performance when utilizing Logistic Regression as the classifier. However, these differences are very small. This indicates that the

synthetic data successfully captures the underlying patterns of the real data reasonably well.

Figure 4.16 shows the ROC curve for CTGAN on the Random Forest Classifier. Since CopulaGAN had similar results, only CTGAN is shown here. All the different classifiers exhibited the same behavior, with the real and synthetic curves always being almost exactly the same, with an AUC between 0.7-0.8. For the Random Forest Classifier depicted below, when training on synthetic data, the AUC of testing on real increased from 0.78 to 0.81. While this is a small uptick in performance, it does suggest that synthetic data is a good representative of the real data. Especially since this behavior was consistent across the other classifiers as well. This suggests that the findings are robust and not specific to a single model.

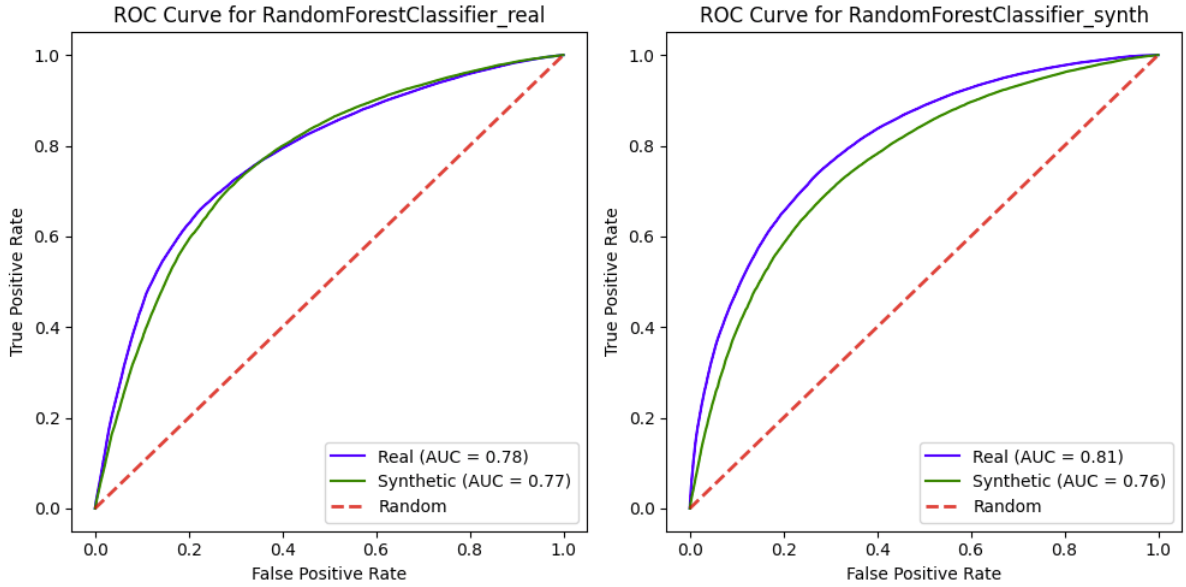


Figure 4.16: ROC plots comparing CTGAN datasets on the *Cardiovascular Disease* dataset. Classifier trained on real data is on the left and trained on synthetic data on the right.

4.3.4 Augmented Data Performance

To generate the augmented dataset, a combination of 50% real data and 100% synthetic data was employed, resulting in an augmented dataset that was 1.5 times larger than the original. This approach was consistently applied to all the datasets.

Classifier	**f1_real**
Logreg	0.6866
RFC	0.7817
MLP	0.7418

Table 4.6: Classifier performance when training and testing on real *Lower Back Pain* data.

	CTGAN		CopulaGAN	
Classifier	**f1_real**	**f1_augmented**	**f1_real**	**f1_augmented**
Logreg	0.6638	0.7973	0.5882	0.7870
RFC	0.7805	0.8540	0.7455	0.8683
MLP	0.7004	0.8104	0.6409	0.7563

Table 4.7: Classifier performance for training on the augmented *Lower Back Pain* data.

As a refresher, Table 4.6 shows how well the Lower Back Pain dataset performs when training and testing on real data in terms of F1 scores. While, table 4.7 shows the scores for training on augmented data and testing on real and augmented data.

Ideally, to see an improvement in performance with augmented data, training on augmented and testing on real should yield better results than training on real and testing on real. Based on the F1 scores, this is not the case. However, it is observed that training on augmented data provides similar performance. For instance, RFC is achieving almost identical results when training on augmented and tested on real for CTGAN-generated data. A notable observation is that when training on augmented data and testing on augmented data, the performance is improved for all the classifiers on all the GAN-generated data. These results can be analyzed further by looking at all the ROC curves for Logistic Regression on CTGAN-generated data:

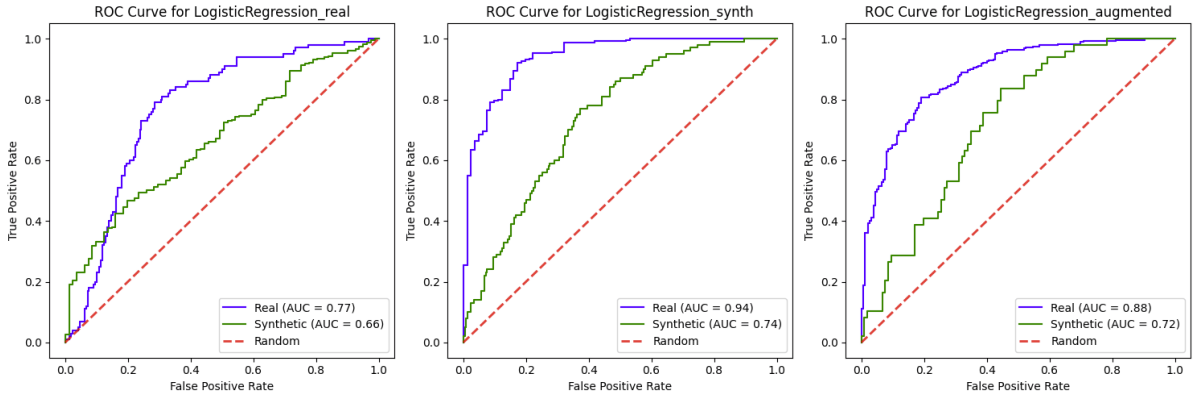


Figure 4.17: All ROC plots for Logistic Regression with CTGAN-generated data based on the *Lower Back Pain* dataset.

From the plots, one can observe that the classifier trained on augmented data performs worse when tested on real data, with an AUC of 0.88, compared to when trained on synthetic data and tested on real data, which has an AUC of 0.94. However, both perform better than training on real data and testing on real data, which has an AUC of 0.77.

As discussed previously, training on synthetic data and testing on real data increased the performance, and it could possibly be due to the fact that the GAN model is managing to generate data that is cleaner than the real data. Therefore, mixing real and synthetic data in the augmentation process is actually decreasing its performance. However, it is important to keep in mind that using augmented data when the

synthetic data has been proven throughout these results to not be a very good representative of the real data, is something to be cautious of.

In the case of the *Obesity* dataset, no noticeable improvement in performance was observed for classifiers trained on augmented data. The F1 scores remained similar to those obtained from training on synthetic data alone, and there was no significant enhancement in the AUC either. Given that the dataset already exhibited high performance when trained on real data alone, the addition of synthetic data resulted in a decrease in performance. Therefore, in this particular scenario, the utilization of augmentation was unnecessary. This can be observed by the following ROC plots:

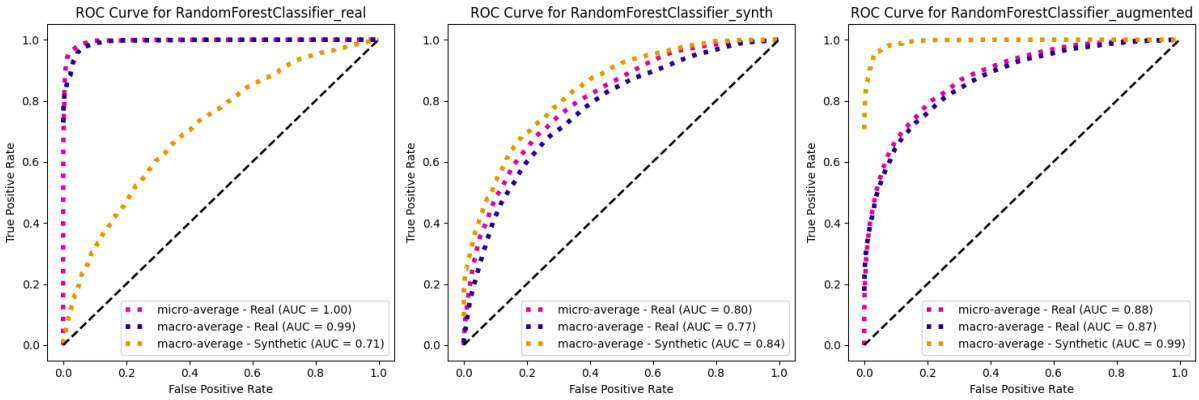


Figure 4.18: All ROC plots for MLP Classifier with CTGAN-generated data based on the *Obesity* dataset.

Lastly, when examining the *Cardiovascular Disease* dataset, no substantial improvements in performance were observed when training on augmented data, as depicted in Figure 4.19. This can be attributed to the fact that the *Cardiovascular Disease* dataset already comprised a substantial number of data points (70,000). It is likely that the complexity of the real data itself was not sufficient to warrant significant performance improvements through the addition of augmented data.

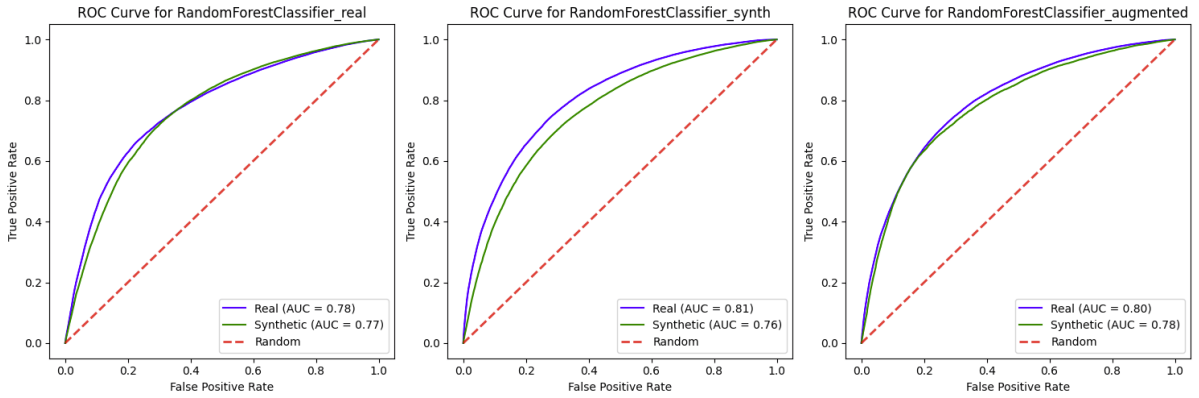


Figure 4.19: All ROC plots for RFC with CTGAN-generated data based on the *Cardiovascular Disease* dataset.

4.4 Preserved Privacy

In this section, the privacy dimension metrics discussed in Section 3.4.3 will be examined. These metrics include the detection of duplicate rows in both the real and

synthetic datasets, as well as exact matches. Furthermore, the nearest neighbor means and standard deviations will also be analyzed.

With the exception of CopulaGAN's *Cardiovascular Disease* dataset, which contained two duplicate rows, none of the models generated synthetic datasets with duplicates. Considering the size of the dataset (70,000 samples), two duplicates are negligible and can be deleted. Moreover, all the orignial datasets, except for *Lower Back Pain*, had some duplicate rows. *Obesity* had 33 duplicates, and *Cardiovascular Disease* had 48. Since the models did not generate duplicate rows, it indicates that they somewhat managed to clean the data. This also suggests that the models did not suffer from mode collapse.

	CTGAN		CopulaGAN	
Dataset	MT_0	$MT_{0.01}$	MT_0	$MT_{0.01}$
Lower Back Pain	0	0	0	0
Obesity	0	0	0	0
Cardiovascular Disease	0	479	2	7061

Table 4.8: Exact Matches for each dataset with different match tolerance percentages.

The number of Exact Matches using different match tolerances is shown in Table 4.8. From the table, it is observed that most of the generated datasets had zero exact matches with the tolerance being 0. This means that there were no rows between the real and fake data that were exactly the same. The exception for this was CopulaGAN for the *Cardiovascular Disease* dataset with 2 matches. Furthermore, setting the tolerance to 1%, increased the matches for CTGAN and CopulaGAN on the *Cardiovascular Disease* to 479 and 7061 identical rows, respectively. This shows that while CopulaGAN had few exact matches, the model did generate data that was more similar to the real data than CTGAN. Given the parallel performance demonstrated by both models in the classifier evaluation, it could suggest that CTGAN may exhibit superior privacy preservation capabilities. However, it all depends on how strict one requires the match-tolerance to be.

Dataset	**CTGAN Dist.**	**CopulaGAN Dist.**
Lower Back Pain	2.4577 ± 0.6120	2.5150 ± 0.6372
Obesity	2.1122 ± 0.4860	2.1329 ± 0.5000
Cardiovascular Disease	0.7173 ± 1.2038	0.5722 ± 1.1761

Table 4.9: Nearest neighbor distances for all the datasets, shown as $mean \pm std$.

Table 4.9 shows the nearest neighbor distances between the means and standard deviations of the real and synthetic data. The magnitude of the mean and standard deviations depends on the number of columns in the dataset, making the metric fair to compare only within the same dataset.

As discussed in Section 3.4.3, one would want a large mean and low standard deviation to manage the trade-off between similarity and privacy. In other words, the bigger the distance between mean and standard deviation, the better privacy. However, too large of a distance means that the synthetic data is too dissimilar to the original dataset and that the synthetic data is of low quality. Considering *Lower Back Pain* datasets has the highest distances, while *Cardiovascular Disease* has the lowest, reflects previous results. Also, CopulaGAN has the lowest distance between the models on the *Cardiovascular Disease* dataset, although the difference is relatively minor. However, for the *Cardiovascular Disease* dataset, the mean is lower than the standard deviations for both models. The high variance between the rows could be a privacy concern, as this could suggest that some rows are more similar than others. Furthermore, a low mean might indicate excessive overfitting to the real data, which could potentially compromise privacy.

Chapter 5

Discussion

In this discussion chapter, a summary of the findings is presented based on the evaluation of synthetic data generated using CTGAN and CopulaGAN. The evaluation focuses on three dimensions: resemblance, classifier evaluation, and privacy. By analyzing the results within each dimension, the main problem statements addressed in this thesis can be discussed.

5.1 Producing High-Fidelity Synthetic Data

While both GAN models had comparable results, it was challenging to determine which one performed better. During the training of GAN models on the three experimental datasets, neither model achieved loss stability on the *Lower Back Pain* or *Obesity* dataset. However, during training on the *Cardiovascular Disease* dataset, both the generator and discriminator losses demonstrated a stabilized trend. It appeared that CopulaGAN achieved slightly better stability compared to CTGAN.

After training the models and generating the synthetic datasets, a comprehensive evaluation was conducted to examine the similarities between the real and synthetic data. The evaluation included analyzing correlation matrices, correlation distances, distribution plots, as well as assessing KSComplements and TVComplements. Notably, CTGAN exhibited a greater ability to capture the underlying correlations and distributions within smaller datasets compared to CopulaGAN. However, as more data was incorporated, the performance gap between the models became less noticeable. The exact reasons behind CTGAN's superior performance on smaller datasets are difficult to determine conclusively, but it suggests that this model may be better suited for such scenarios. It is important to note that other factors, such as the unique characteristics of the data, could have influenced these results. Consequently, it is advisable to explore different models to determine the most suitable one for specific datasets.

The evaluation of resemblance in the synthetic datasets revealed varying results across the three datasets, with the synthetic *Lower Back Pain* datasets struggling the most in replicating the real data's complexities. The primary contributing factors to this issue were the small dataset size of only 310 data points and its highly imbalanced nature. In comparison, the synthetic *Obesity* datasets displayed slightly better results but still had room for improvement. The synthetic *Cardiovascular Disease* datasets

demonstrated the best resemblance to real data among the three datasets, as evidenced by the distribution plots and correlation matrices, which showed that both models successfully captured the complexities of the real data.

These findings suggest that GAN models might require a larger volume of data than what was available in the *Lower Back Pain* dataset to effectively learn underlying patterns, and the poor quality of this dataset further worsened the problem. In contrast, the *Obesity* dataset, consisting of only 2111 data points, performed significantly better. This indicates that sufficient variability and limited noise in real data can lead to more successful GAN-generated synthetic datasets, even with smaller sample sizes. However, while the synthetic data generated from the Obesity dataset captured many of the real data's complexities, it still fell short of representing the full range of intricacies found in the real data. The impressive results obtained from the *Cardiovascular Disease* dataset, with its large size of 70,000 data points, further emphasize the importance of having a substantial volume of high-quality data for GAN models to effectively capture and reproduce the intricate patterns present in real datasets.

5.2 Comparing Classifier Performance

Training various classifiers on synthetic datasets provided valuable insights into the effectiveness of synthetic data in capturing the underlying distributions of real data. Ideally, classifiers trained on synthetic data and tested on real data should exhibit comparable performance to classifiers trained and tested solely on real data. The SynthEval framework, developed for this thesis, facilitated the evaluation of synthetic data's predictive power in comparison to real data, addressing the first problem statement of this thesis.

Classifier performance often reflects how well synthetic data capture real data complexities. For the *Obesity* dataset, the importance of having quality data was emphasized, as certain target classes with insufficient data caused difficulties in predicting those specific classes. Moreover, while training on real data and testing on real data resulted in high performance, training on the lower-quality synthetic dataset led to worse performance. In contrast, The *Cardiovascular Disease* dataset performed as well as real data in making predictions, indicating successful capture of the underlying patterns in the real data.

Some intriguing observations emerged from the SynthEval Framework. In some cases, synthetic data outperformed real data in classifier performance, yielding better results when training on synthetic data and testing on real data compared to training and testing on real data. This occurred for the *Lower Back Pain* dataset and occasionally for the *Cardiovascular Disease* dataset, although the performance increase was minimal in the latter case. This outcome could be explained by GAN models generating less noisy data with more variation than the original data. Consequently, synthetic data might contain fewer errors and inconsistencies, enabling prediction models to extract patterns and make accurate predictions more easily.

However, this may also lead to synthetic data not accurately representing the true distribution of the real data, resulting in biased or unreliable predictions. Synthetic data might not capture outliers or biases present in real data, which could lead to better performance on synthetic data but also problems if certain patterns are missed.

For instance, the *Lower Back Pain* dataset consistently showed that synthetic data was not a good representation of the real data, yet the performance increased from 0.77 to 0.94 when training on synthetic data and testing on real data. Therefore, it is crucial to ensure synthetic data reliability by verifying that it captures data complexities before using it in decision-making or analysis applications, as unreliable synthetic data can lead to incorrect diagnoses, treatments, or medical decisions.

5.3 Enhancing Classifier Performance through Synthetic Data Augmentation

The second problem statement of this thesis aimed to investigate the extent to which augmenting real data with synthetic data can enhance classifier performance. To address this, the SynthEval framework was employed to compare the performance of classifiers using real, synthetic, and augmented data, in order to assess whether any improvements in performance could be observed.

However, a notable increase in classifier performance was observed only within the *Lower Back Pain* dataset. For example, with the Logistic Regression model, when trained on augmented data and tested on real data, the AUC reached 0.88. This represents a significant improvement compared to training and testing on real data, with an AUC of 0.77. It is important to highlight that the performance of the augmented data was lower than when training on pure synthetic data and testing on real data. In this specific case, it suggests that the synthetic data alone may have captured certain patterns more effectively than the combination of real and synthetic data, or that the synthetic data introduced less noise, ultimately leading to enhanced classifier performance.

In contrast, the *Obesity* dataset already exhibited strong results when using real data. However, since the synthetic data could not match the performance level of the real data, incorporating it into an augmented dataset did not yield any improvements in classifier performance. Additionally, the *Cardiovascular Disease* dataset successfully generated synthetic data with performance comparable to that of the real data. Despite this, augmenting the real data did not enhance the performance. This is likely because the dataset was already large, and any performance limitations were not due to missing data points, but rather the inherent complexity of the data.

This situation presents a catch-22 because data augmentation is typically more valuable when dealing with data that is not sufficient enough for training, yet GAN models require a sufficient amount of data to generate high-quality synthetic data. Since the GAN models had difficulties capturing the underlying distributions and correlations of the smaller datasets, the generated synthetic datasets were of lower quality. Using datasets of lower quality to augment data in healthcare applications, as previously discussed, is problematic. Therefore, it is essential to carefully evaluate the quality of synthetic data generated from smaller datasets and consider alternative data augmentation techniques or other approaches to improve classifier performance when dealing with insufficient data. This highlights the importance of continued research and development in generating high-quality synthetic data, especially for smaller or more complex datasets.

5.4 Balancing Privacy and Similarity

The results suggest that it is indeed possible to create synthetic data that closely resembles the real data, as demonstrated by the successful replication of the *Cardiovascular Disease* dataset. However, it is crucial to be mindful of the potential privacy concerns associated with generating data that is overly similar to the original. This aspect aligns with the objective of the third problem statement addressed in this thesis.

Considering the low amount of duplicates and exact matches between the real and synthetic data, it does seem that the models manage to preserve privacy somewhat. However, raising the tolerance level for the *Cardiovascular Disease* dataset results in a noticeable increase in exact matches, particularly with CopulaGAN. This suggests that a more significant balance between data privacy and similarity might be necessary, especially when handling highly sensitive information.

The *Lower Back Pain* dataset seemed to have few privacy issues, however the relevancy of this is questionable, as the generated data was of low quality. The *Obesity* dataset also seemed to have managed to preserve the privacy relatively well, with no exact matches in the synthetic data and a high mean and low standard deviation from the nearest neighbor distances. Furthermore, the *Cardiovascular Disease* dataset achieved the highest performance of all the datasets, but there also seemed to be some signs of overfitting. Although the privacy metrics showed fairly acceptable results, the dataset's high variance and lower mean might raise concerns. This indicates that the models have over-learned from the data, making the synthetic data more vulnerable to potential information leakage. However, it is important to note that using nearest neighbor as the sole privacy metric may not be the most reliable approach. Further research and exploration of additional privacy metrics are necessary in this field to reach a definitive conclusion regarding the effectiveness of privacy preservation.

5.5 Limitations

The study only evaluated two GAN models (CTGAN and CopulaGAN) on a limited number of datasets. While the results are promising in some cases, it is unclear how well these models would perform on other datasets with different characteristics. In some cases, some of the data used were not complex enough to fully explore different areas of analysis. For instance, the *Cardiovascular Disease* dataset lacked the necessary complexity to accurately test the effectiveness of augmenting real data with synthetic data. Additionally, the data used in this study only comprised of continuous and discrete data types. As a result, the GAN models were evaluated solely on these specific data types, and their performance on other data types was not assessed.

Furthermore, the data used in the study was obtained from open-source websites. As a result, the data has undergone pre-processing, which raises concerns regarding its validity. It is recommended to utilize authentic data directly sourced from hospitals or laboratories, as such data is typically not anonymized or altered.

While commonly used metrics for privacy were employed in this study, there may be other metrics that are more appropriate for evaluating privacy in healthcare datasets. Although they provide a general overview of how well the models can preserve privacy, they do not offer a comprehensive enough analysis to conclude if the data is really secure.

5.6 Future work

Throughout this thesis, the evaluation techniques employed have yielded encouraging findings in generating synthetic datasets that accurately capture the inherent patterns of real data. Nevertheless, there remain intriguing avenues for future research that warrant exploration and attention. These areas hold potential for further advancements in the field of synthetic data generation.

Investigating the impact of different generation algorithms or architectures on the quality of synthetic datasets is an area worth exploring. This thesis focused solely on the use of GAN models to generate synthetic data, but future work could explore alternative methods for STDG, such as the use of auto-encoders or RNNs.

Exploring the performance of different GAN models on more complex datasets would be an interesting area of future research. This could involve data with more features and complex relationships between them. Additionally, testing on more complex data types such as time-series data could be considered. Furthermore, exploring the effectiveness of synthetic data for applications beyond classification tasks would also be valuable. This could include clustering, time-series analysis, or forecasting.

This thesis established a framework specifically designed for evaluating prediction tasks with medical data, incorporating metrics such as F1 scores, AUC, and ROC curves. However, the range of evaluation metrics available could potentially be expanded to provide a broader and more comprehensive assessment. By incorporating additional metrics tailored to the unique characteristics of medical data, the evaluation process can be further improved.

While the primary focus of this research was to evaluate data quality and performance, future work could try and incorporate more privacy metrics and compare performance and quality against these metrics. It is worth investigating whether maintaining privacy could have a negative impact on performance and where the optimal balance between these two dimensions lies. One potential approach to explore this further is to generate synthetic data using more sensitive or seemingly sensitive data to evaluate the performance of different inference attack methods or other privacy metrics discussed in related literature. Another option could be to investigate other privacy metrics for STDG, considering this is an under-researched area.

Finally, conducting user studies or real-world evaluations to assess the practical applicability of synthetic data would be highly valuable. Given the concerns surrounding data bias and potential erroneous predictions, gathering feedback from domain experts or end-users, especially in clinical settings, can significantly enhance the reliability and trustworthiness of the generated data. This feedback could contribute to mitigating the risk of bias and inaccuracies that may arise in synthetic data.

Chapter 6

Conclusion

The healthcare sector presents vast opportunities for the integration and advancement of artificial intelligence. Leveraging the capabilities of AI can revolutionize diagnostic and prognostic modeling, streamline patient risk assessment, and enhance clinical decision support, among other notable applications. Despite these promising prospects, the progress in healthcare AI development is impeded by strict data protection laws and regulations safeguarding sensitive patient information, which restrict researchers' access to authentic medical datasets. Nevertheless, synthetic data emerges as a promising solution to circumvent these challenges, as it offers the potential to generate data that closely resembles real-world datasets without compromising individual privacy.

The primary objective of this thesis was to explore the generation of synthetic tabular healthcare data by employing several GAN models, thereby assessing the potential and significance of synthetic tabular data generation within the healthcare industry. The two GAN models selected, CTGAN and CopulaGAN, were chosen based on a comprehensive systematic review of prevalent tabular GAN models. Subsequently, three diverse healthcare datasets of varying sizes and complexities were selected: *Lower Back Pain Symptoms*, *Estimation of obesity levels*, and *Cardiovascular Disease Prediction*. To optimize performance, the two GAN models were trained using distinct parameters for each dataset. Convergence of the GAN models was only observed when training on the *Cardiovascular Disease* dataset.

Upon generating synthetic data using the GAN models based on the chosen datasets, a thorough evaluation process was conducted to assess the synthetic data from three distinct dimensions: resemblance, classifier performance, and privacy. The metrics employed within each dimension were aligned with those commonly used in related studies. To evaluate resemblance, univariate statistical characteristics were compared using simple statistical checks, alongside column correlation and feature distribution comparisons.

A custom-made framework, "SynthEval," was developed to assess the synthetic data against the real data in terms of classifier performance. This framework aimed to evaluate real, synthetic, and augmented data relative to one another. SynthEval consisted of three phases: the first phase involved training a classifier on real data and evaluating it on both real and synthetic data; the second phase trained the classifier

on synthetic data and evaluated it on both synthetic and real data; and the third phase focused on augmented data, combining real and synthetic data. The third phase sought to determine whether incorporating synthetic data alongside real data would enhance the performance of the classifier. Through this framework, a comprehensive evaluation of various classifiers and data was achieved, with results presented as F1 scores and ROC curves. For this thesis, three classifiers were employed within the SynthEval framework: Logistic Regression, Random Forest, and MLPClassifier. Finally, privacy metrics such as duplicates, exact matches, and nearest neighbor distances were utilized to determine whether the GAN models effectively generated synthetic data without compromising privacy.

The results revealed that synthetic data based on the *Lower Back Pain* dataset, the smallest dataset with 310 datapoints, faced the greatest difficulty in capturing the real data's underlying patterns. The *Obesity* dataset demonstrated moderate success, though there remained room for improvement. In contrast, the largest dataset, *Cardiovascular Disease*, with 70,000 datapoints, appeared to successfully capture correlations and distributions consistent with the real data. As a result, the GAN-generated datasets exhibited predictive power equivalent to the real data. While classifier performance declined for the synthetic *Obesity* dataset, the *Lower Back Pain* dataset performed better on synthetic data than the real data, suggesting cleaner synthetic data. This dataset also exhibited improved performance when augmented data was used.

In terms of privacy, the synthetic *Lower Back Pain* and *Obesity* datasets exhibited the fewest concerns; however, this may be attributed to their limited representation of real data. Although the GAN models generated high-quality synthetic data for the *Cardiovascular Disease* dataset, the privacy evaluation raised concerns about potential overfitting, which could indicate that the models over-learned from the data. A more in-depth privacy assessment is necessary to draw a definitive conclusion on this matter.

In conclusion, this thesis has demonstrated the feasibility of generating synthetic tabular healthcare data using GAN models, showcasing its potential to mitigate challenges posed by data protection laws and regulations. Although some datasets yielded better results than others, the overall findings highlight the promise of synthetic data as a viable alternative to real data in certain circumstances. Moving forward, future research should prioritize refining GAN models and evaluation methodologies, utilizing more complex data for synthetic data generation, and conducting in-depth privacy assessments. These efforts will help ensure responsible and ethical advancement of synthetic data applications within healthcare.

.

Bibliography

[1] Santiago Gomez Paz. Demystifying the ctgan loss function, synthetic data modeling. https://github.com/sdv-dev/SDV/discussions/980, 2021. Accessed: [28.04.23].

[2] Matthew Ventresca, Holger J Schünemann, Fergus Macbeth, Mike Clarke, Lehana Thabane, Gareth Griffiths, Simon Noble, David Garcia, Maura Marcucci, Alfonso Iorio, et al. Obtaining and managing data sets for individual participant data meta-analysis: scoping review and practical guide. *BMC Medical Research Methodology*, 20(1):1–18, 2020.

[3] João Coutinho-Almeida, Pedro Pereira Rodrigues, and Ricardo João Cruz-Correia. Gans for tabular healthcare data generation: A review on utility and privacy. In Carlos Soares and Luis Torgo, editors, *Discovery Science*, pages 282–291, Cham, 2021. Springer International Publishing. ISBN 978-3-030-88942-5.

[4] Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. *Future healthcare journal*, 6(2):94, 2019.

[5] Richard J Chen, Ming Y Lu, Tiffany Y Chen, Drew FK Williamson, and Faisal Mahmood. Synthetic data in machine learning for medicine and healthcare. *Nature Biomedical Engineering*, 5(6):493–497, 2021.

[6] Tim Hulsen. Sharing is caring—data sharing initiatives in healthcare. *International Journal of Environmental Research and Public Health*, 17(9), 2020. ISSN 1660-4601. doi: 10.3390/ijerph17093046. URL https://www.mdpi.com/1660-4601/17/9/3046.

[7] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 289–293, 2018. doi: 10.1109/ISBI.2018.8363576.

[8] Nikita Jaipuria, Xianling Zhang, Rohan Bhasin, Mayar Arafa, Punarjay Chakravarty, Shubham Shrivastava, Sagar Manglani, and Vidya N Murali. Deflating dataset bias using synthetic data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 772–773, 2020.

[9] Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings 5*, pages 1–19. Springer, 2008.

[10] Khaled El Emam and Fida Kamal Dankar. Protecting Privacy Using k-Anonymity. *Journal of the American Medical Informatics Association*, 15(5):627–637, 09 2008. ISSN 1067-5027. doi: 10.1197/jamia.M2716. URL https://doi.org/10.1197/jamia.M2716.

[11] James H Boyd, Sean M Randall, Anna M Ferrante, Jacqueline K Bauer, Adrian P Brown, and James B Semmens. Technical challenges of providing record linkage services for research. *BMC medical informatics and decision making*, 14(1):1–9, 2014.

[12] Roderick JA Little et al. Statistical analysis of masked data. *JOURNAL OF OFFICIAL STATISTICS-STOCKHOLM-*, 9:407–407, 1993.

[13] Donald B Rubin. Statistical disclosure limitation. *Journal of official Statistics*, 9(2): 461–468, 1993.

[14] Gillian M Raab, Beata Nowok, and Chris Dibben. Practical data synthesis for large samples. *Journal of Privacy and Confidentiality*, 7(3):67–97, 2016.

[15] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.04.053. URL https://www.sciencedirect.com/science/article/pii/S0925231222004349.

[16] Christoph Baur, Shadi Albarqouni, and Nassir Navab. Generating highly realistic images of skin lesions with gans, 2018.

[17] Yash Karbhari, Arpan Basu, Zong Woo Geem, Gi-Tae Han, and Ram Sarkar. Generation of synthetic chest x-ray images and detection of covid-19: A deep learning based approach. *Diagnostics*, 11(5):895, 2021.

[18] Dan Yoon, Hyoun-Joong Kong, Byeong Soo Kim, Woo Sang Cho, Jung Chan Lee, Minwoo Cho, Min Hyuk Lim, Sun Young Yang, Seon Hee Lim, Jooyoung Lee, et al. Colonoscopic image synthesis with generative adversarial network for enhanced detection of sessile serrated lesions using convolutional neural network. *Scientific reports*, 12(1):261, 2022.

[19] Ghadeer Ghosheh, Jin Li, and Tingting Zhu. A review of generative adversarial networks for electronic health records: applications, evaluation measures and data sources. *arXiv preprint arXiv:2203.07018*, 2022.

[20] Anders Krogh. What are artificial neural networks? *Nature Biotechnology*, 26(2): 195–197, 2008. ISSN 1087-0156.

[21] Sundaramoorthy Rajasekaran and GA Vijayalakshmi Pai. *Neural networks, fuzzy systems and evolutionary algorithms: Synthesis and applications*. PHI Learning Pvt. Ltd., 2017.

[22] Muhammad Uzair and Noreen Jamil. Effects of hidden layers on the efficiency of neural networks. In *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pages 1–6, 2020. doi: 10.1109/INMIC50486.2020.9318195.

[23] Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[24] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[25] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[26] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[28] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841, 2018. doi: 10.1109/CCDC.2018.8407425.

[29] Will Koehrsen. Overfitting vs. underfitting: A complete example. *Towards Data Science*, pages 1–12, 2018.

[30] Feng Li, Jacek M. Zurada, Yan Liu, and Wei Wu. Input layer regularization of multilayer feedforward neural networks. *IEEE Access*, 5:10979–10985, 2017. doi: 10.1109/ACCESS.2017.2713389.

[31] Ekachai Phaisangittisagul. An analysis of the regularization between l2 and dropout in single hidden layer neural network. In *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pages 174–179, 2016. doi: 10.1109/ISMS.2016.14.

[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

[33] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. *Supervised Learning*, pages 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-75171-7. doi: 10.1007/978-3-540-75171-7_2. URL https://doi.org/10.1007/978-3-540-75171-7_2.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, oct 2020. ISSN 0001-0782. doi: 10.1145/3422622. URL https://doi.org/10.1145/3422622.

[35] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018. doi: 10.1109/MSP.2017.2765202.

[36] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[37] Divya Saxena and Jiannong Cao. Generative adversarial networks (gans): Challenges, solutions, and future directions, 2020. URL https://arxiv.org/abs/2005.00065.

[38] Alvaro Figueira and Bruno Vaz. Survey on synthetic data generation, evaluation methods and gans. *Mathematics*, 10(15), 2022. ISSN 2227-7390. doi: 10.3390/math10152733. URL https://www.mdpi.com/2227-7390/10/15/2733.

[39] Afia Sajeeda and B M Mainul Hossain. Exploring generative adversarial networks and adversarial training. *International Journal of Cognitive Computing in Engineering*, 3:78–89, 2022. ISSN 2666-3074. doi: https://doi.org/10.1016/j.ijcce.2022.03.002. URL https://www.sciencedirect.com/science/article/pii/S2666307422000080.

[40] Penny Chong, Lukas Ruff, Marius Kloft, and Alexander Binder. Simple and effective prevention of mode collapse in deep one-class classification. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2020. doi: 10.1109/ijcnn48605.2020.9207209. URL https://doi.org/10.1109%2Fijcnn48605.2020.9207209.

[41] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[42] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[43] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class project for Stanford CS231N: convolutional neural networks for visual recognition, Winter semester*, 2014(5):2, 2014.

[44] Vu Nguyen, Tomas F. Yago Vicente, Maozheng Zhao, Minh Hoai, and Dimitris Samaras. Shadow detection with conditional generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4520–4528, 2017. doi: 10.1109/ICCV.2017.483.

[45] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[46] Jonas Teuwen and Nikita Moriakov. Chapter 20 - convolutional neural networks. In S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger, editors, *Handbook of Medical Image Computing and Computer Assisted Intervention*, The Elsevier and MICCAI Society Book Series, pages 481–501. Academic Press, 2020. ISBN 978-0-12-816176-0. doi: https://doi.org/10.1016/B978-0-12-816176-0.00025-9. URL https://www.sciencedirect.com/science/article/pii/B9780128161760000259.

[47] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[48] Jakub Langr and Vladimir Bok. *GANs in action: deep learning with generative adversarial networks*. Manning, 2019.

[49] William Fedus*, Mihaela Rosca*, Balaji Lakshminarayanan, Andrew M. Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByQpn1ZA-.

[50] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf.

[51] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Standardised metrics and methods for synthetic tabular data evaluation. *Preprint at https://doi. org/10.36227/techrxiv*, 16610896:v1, 2021.

[52] Clara García-Vicente, David Chushig-Muzo, Inmaculada Mora-Jiménez, Himar Fabelo, Inger Torhild Gram, Maja-Lisa Løchen, Conceição Granja, and Cristina Soguero-Ruiz. Evaluation of synthetic categorical data generation techniques for predicting cardiovascular diseases and post-hoc interpretability of the risk factors. *Applied Sciences*, 13(7), 2023. ISSN 2076-3417. doi: 10.3390/app13074119. URL https://www.mdpi.com/2076-3417/13/7/4119.

[53] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In Finale Doshi-Velez, Jim Fackler, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 of *Proceedings of Machine Learning Research*, pages 286–305. PMLR, 18–19 Aug 2017. URL https://proceedings.mlr.press/v68/choi17a.html.

[54] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 2019.

[55] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-seeking generative adversarial networks, 2018.

[56] Ramiro Camino, Christian Hammerschmidt, and Radu State. Generating multi-categorical samples with generative adversarial networks, 2018.

[57] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P. Bennett. Generation and evaluation of privacy preserving synthetic health data. *Neurocomputing*, 416:244–255, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.12.136. URL https://www.sciencedirect.com/science/article/pii/S0925231220305117.

[58] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, jun 2018. doi: 10.14778/3231751.3231757. URL https://doi.org/10.14778%2F3231751.3231757.

[59] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks, 2018.

[60] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf.

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

[62] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.

[63] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017.

[64] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training, 2016.

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

[66] Amirarsalan Rajabi and Ozlem Ozmen Garibay. Tabfairgan: Fair tabular data generation with generative adversarial networks. *Machine Learning and Knowledge Extraction*, 4(2):488–501, 2022. ISSN 2504-4990. doi: 10.3390/make4020022. URL https://www.mdpi.com/2504-4990/4/2/22.

[67] Stavroula Bourou, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. A review of tabular data synthesis using gans on an ids dataset. *Information*, 12(9), 2021. ISSN 2078-2489. doi: 10.3390/info12090375. URL https://www.mdpi.com/2078-2489/12/9/375.

[68] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2019.

[69] Lu Wang, Wei Zhang, and Xiaofeng He. Continuous patient-centric sequence generation via sequentially coupled adversarial learning. In *Database Systems for Advanced Applications: 24th International Conference, DASFAA 2019, Chiang Mai, Thailand, April 22–25, 2019, Proceedings, Part II 24*, pages 36–52. Springer, 2019.

[70] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20,

page 343–362, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370899. doi: 10.1145/3372297.3417238. URL https://doi.org/10.1145/3372297.3417238.

[71] Marmar Orooji, Seyedeh Shaghayegh Rabbanian, and Gerald M Knapp. Flexible adversary disclosure risk measure for identity and attribute disclosure attacks. *International Journal of Information Security*, pages 1–15, 2023.

[72] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[73] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2017. doi: 10.1109/SP.2017.41.

[74] Fabio Mendoza Palechor and Alexis de la Hoz Manotas. Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from colombia, peru and mexico. *Data in brief*, 25:104344, 2019.

[75] J. Tacq. The normal distribution and its applications. In Penelope Peterson, Eva Baker, and Barry McGaw, editors, *International Encyclopedia of Education (Third Edition)*, pages 467–473. Elsevier, Oxford, third edition edition, 2010. ISBN 978-0-08-044894-7. doi: https://doi.org/10.1016/B978-0-08-044894-7.01563-3. URL https://www.sciencedirect.com/science/article/pii/B9780080448947015633.

[76] Roger B Nelsen. *An introduction to copulas*. Springer science & business media, 2007.

[77] Synthetic data metrics, 10 2022. URL https://docs.sdv.dev/sdmetrics/. Version 0.8.0.

[78] Bauke Brenninkmeijer, A de Vries, E Marchiori, and Youri Hille. *On the generation and evaluation of tabular data using GANs*. PhD thesis, Radboud University Nijmegen, The Netherlands, 2019.

[79] Charles E. Metz. Basic principles of roc analysis. *Seminars in Nuclear Medicine*, 8(4):283–298, 1978. ISSN 0001-2998. doi: https://doi.org/10.1016/S0001-2998(78)80014-2. URL https://www.sciencedirect.com/science/article/pii/S0001299878800142.

[80] Plotting receiver operating characteristic (roc) curves. https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html, n.d. Accessed on May 4, 2023.

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

[82] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Randomforestclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2021. Accessed: 2023-05-10.

[84] Shuyin Xia, Zhongyang Xiong, Yueguo Luo, WeiXu, and Guanghua Zhang. Effectiveness of the euclidean distance in high dimensional spaces. *Optik*, 126 (24):5614–5619, 2015. ISSN 0030-4026. doi: https://doi.org/10.1016/j.ijleo.2015.09.093. URL https://www.sciencedirect.com/science/article/pii/S0030402615011493.

Appendix A: Lower Back Pain Symptoms

A.1 Implementation

Github Repository with all the implemented code, results and datasets used for this thesis can be found at: https://github.com/mareped/STDG

A.2 Basic Statistical Check

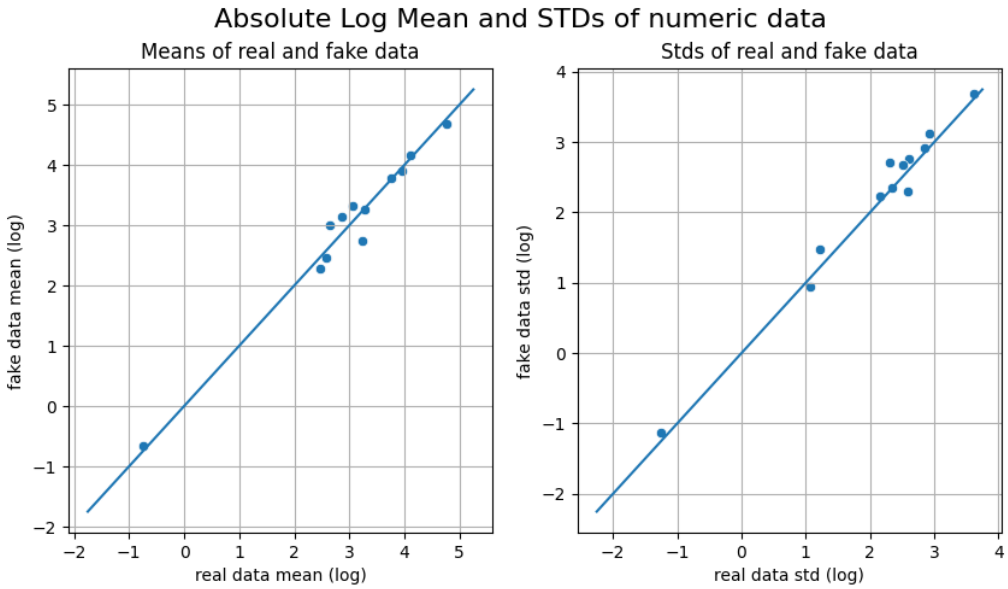


Figure A.1: Absolute Log Mean and STD of numeric data for *Lower Back Pain* dataset generated by CopulaGAN.

A.3 Column Distributions

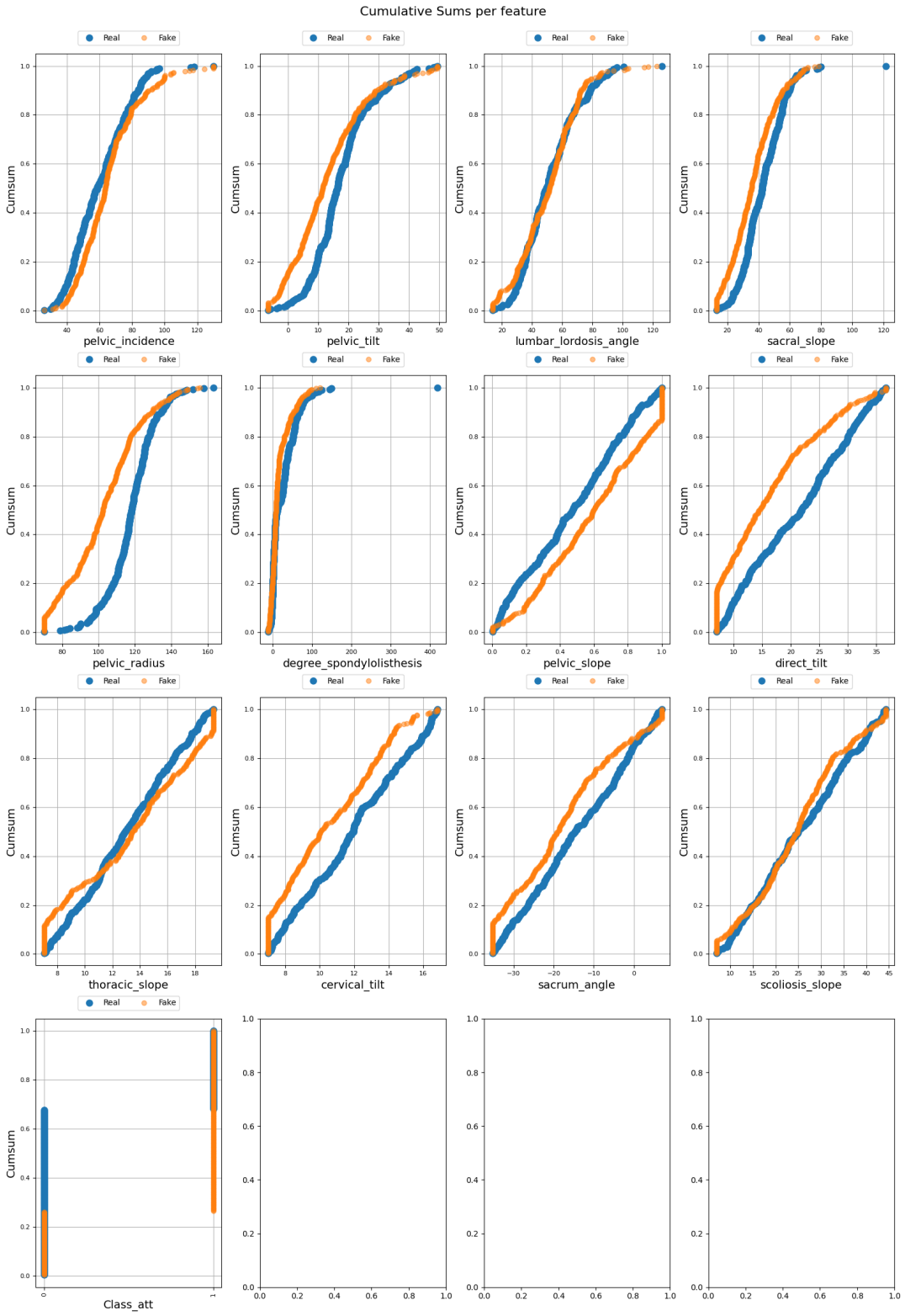


Figure A.2: Cumulative sums of each feature in the *Lower Back Pain* dataset for CTGAN.

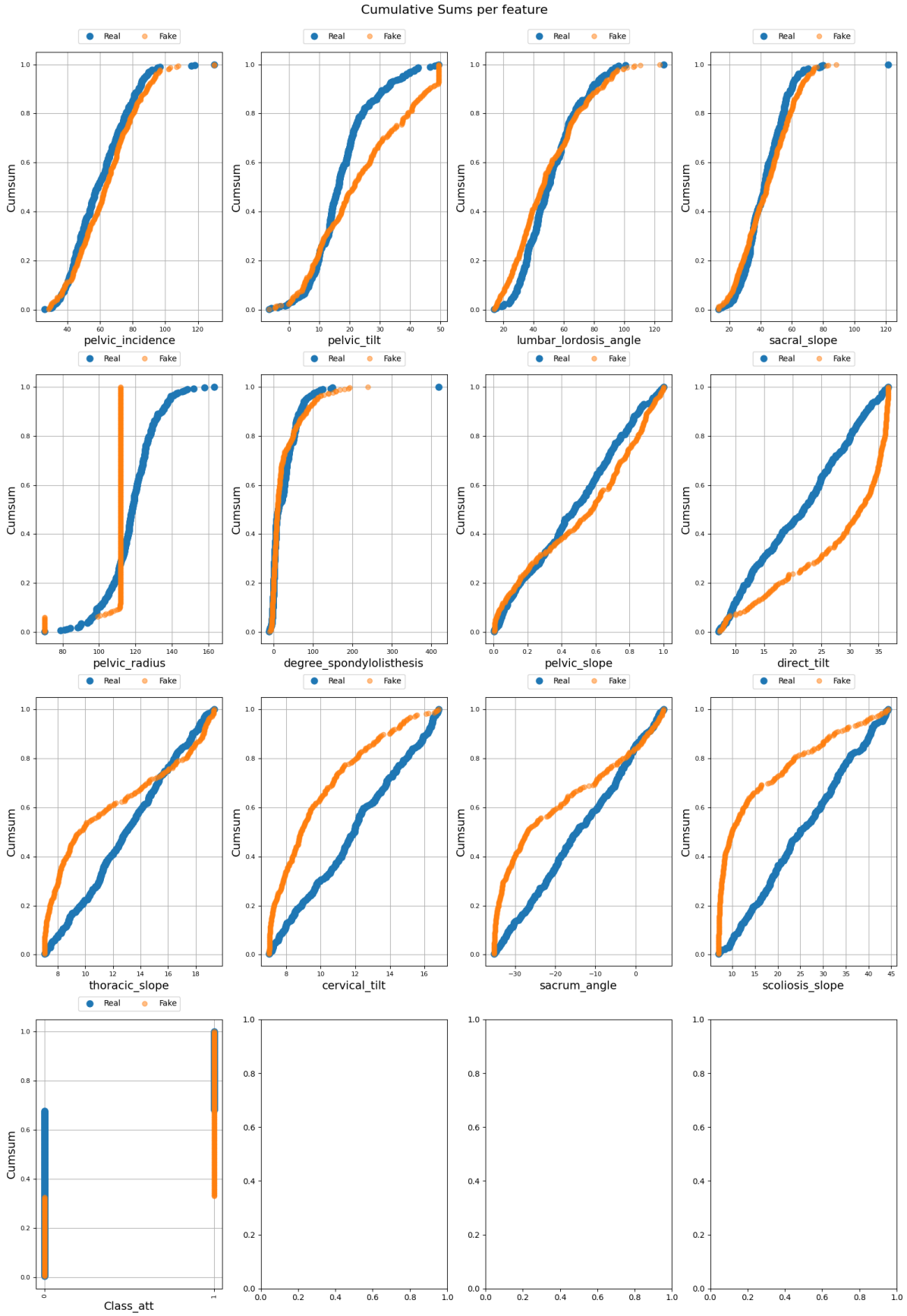


Figure A.3: Cumulative sums of each feature in the *Lower Back Pain* dataset for CopulaGAN.

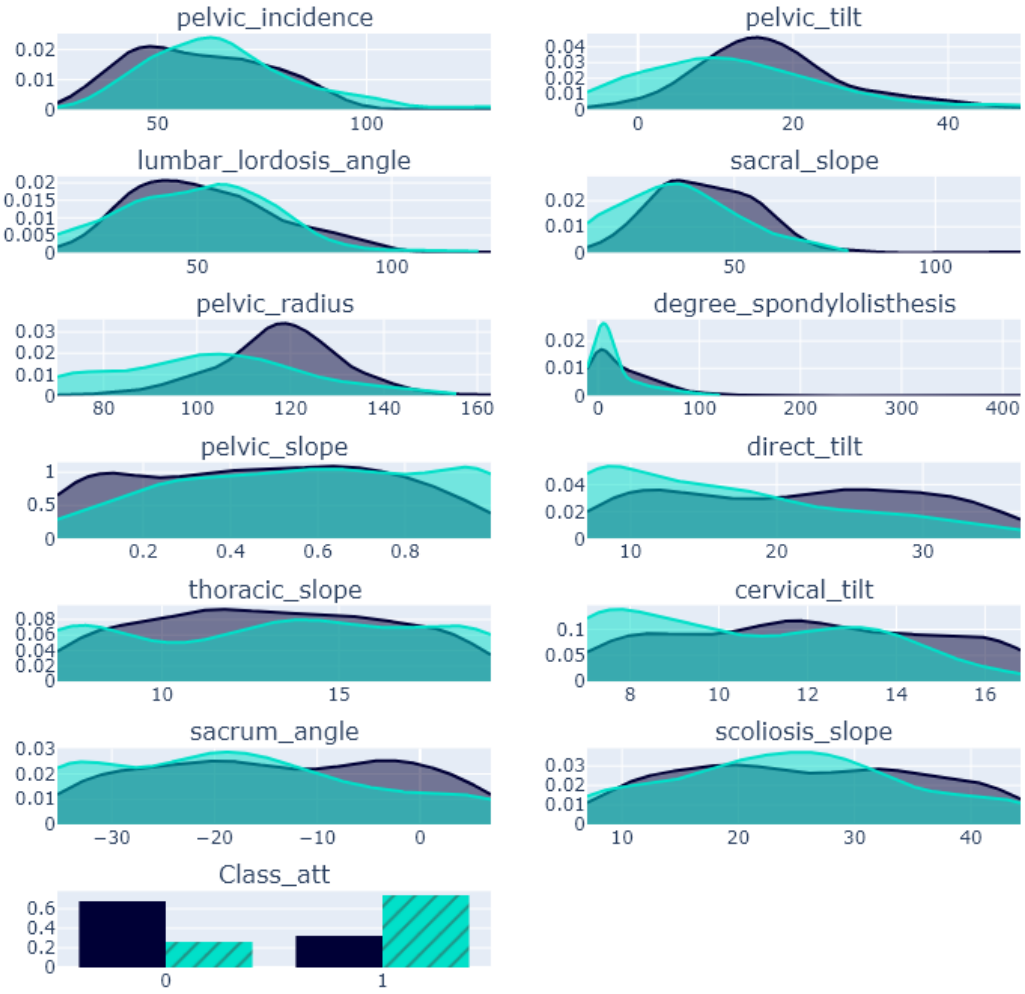


Figure A.4: Feature shape comparison of each column in the *Lower Back Pain* dataset for CTGAN.

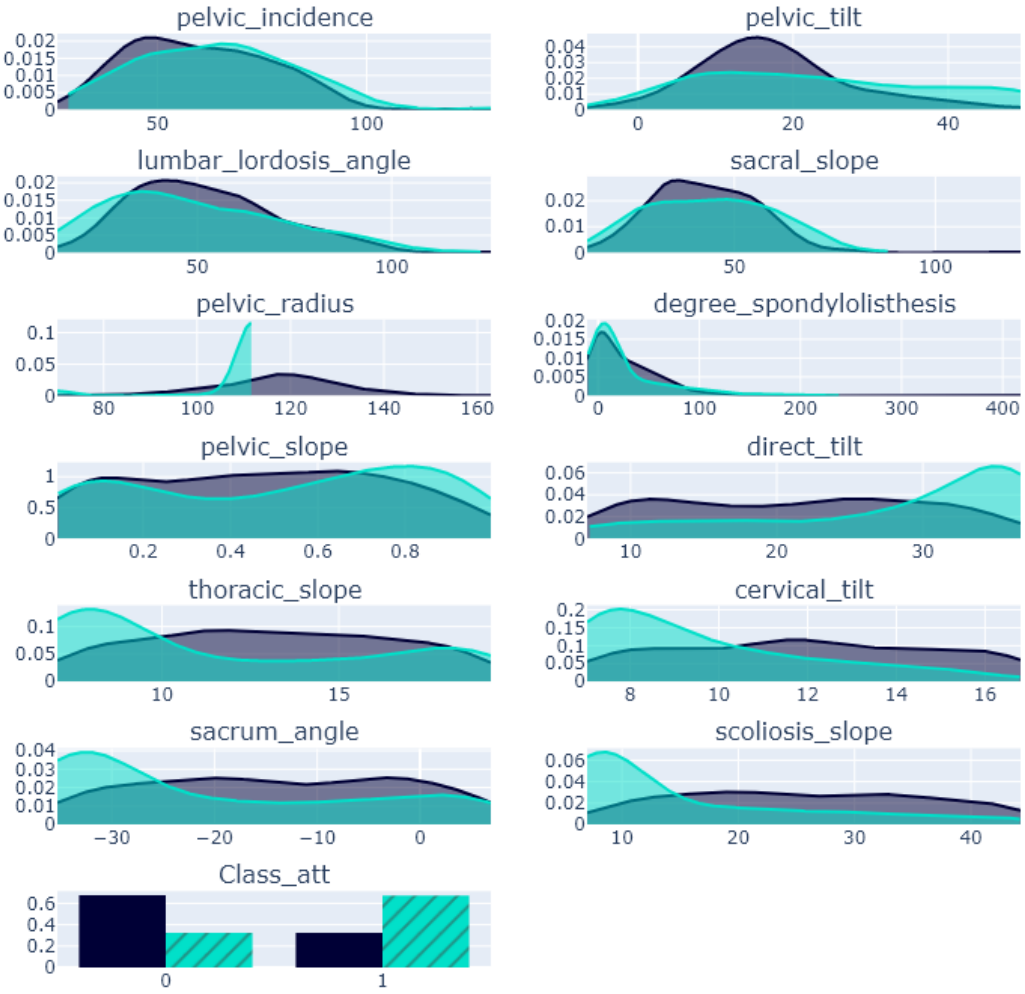


Figure A.5: Feature shape comparison of each column in the *Lower Back Pain* dataset for CopulaGAN.

A.4 Classifier Evaluation

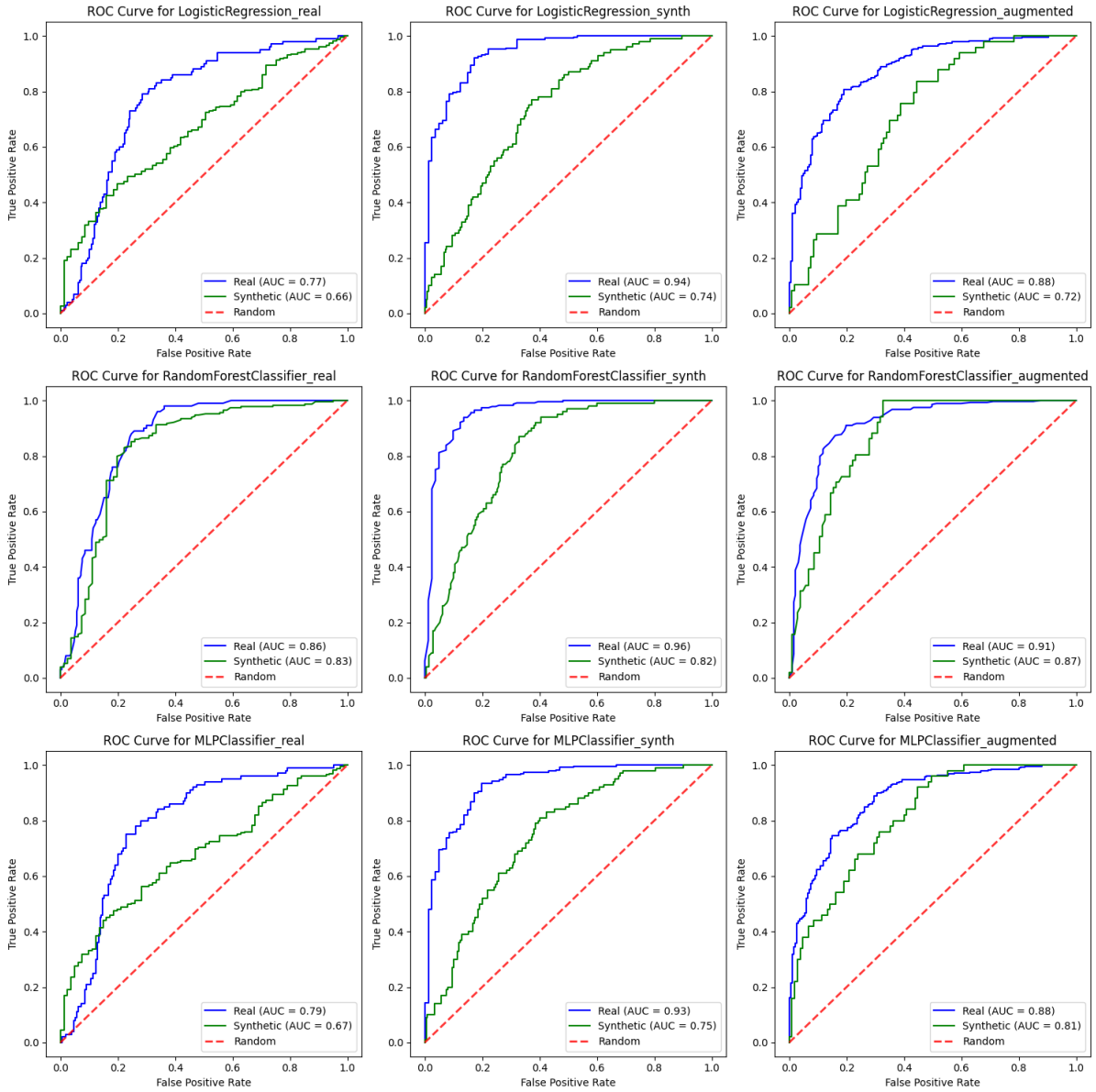


Figure A.6: All ROC curves for CTGAN's *Lower Back Pain* dataset.

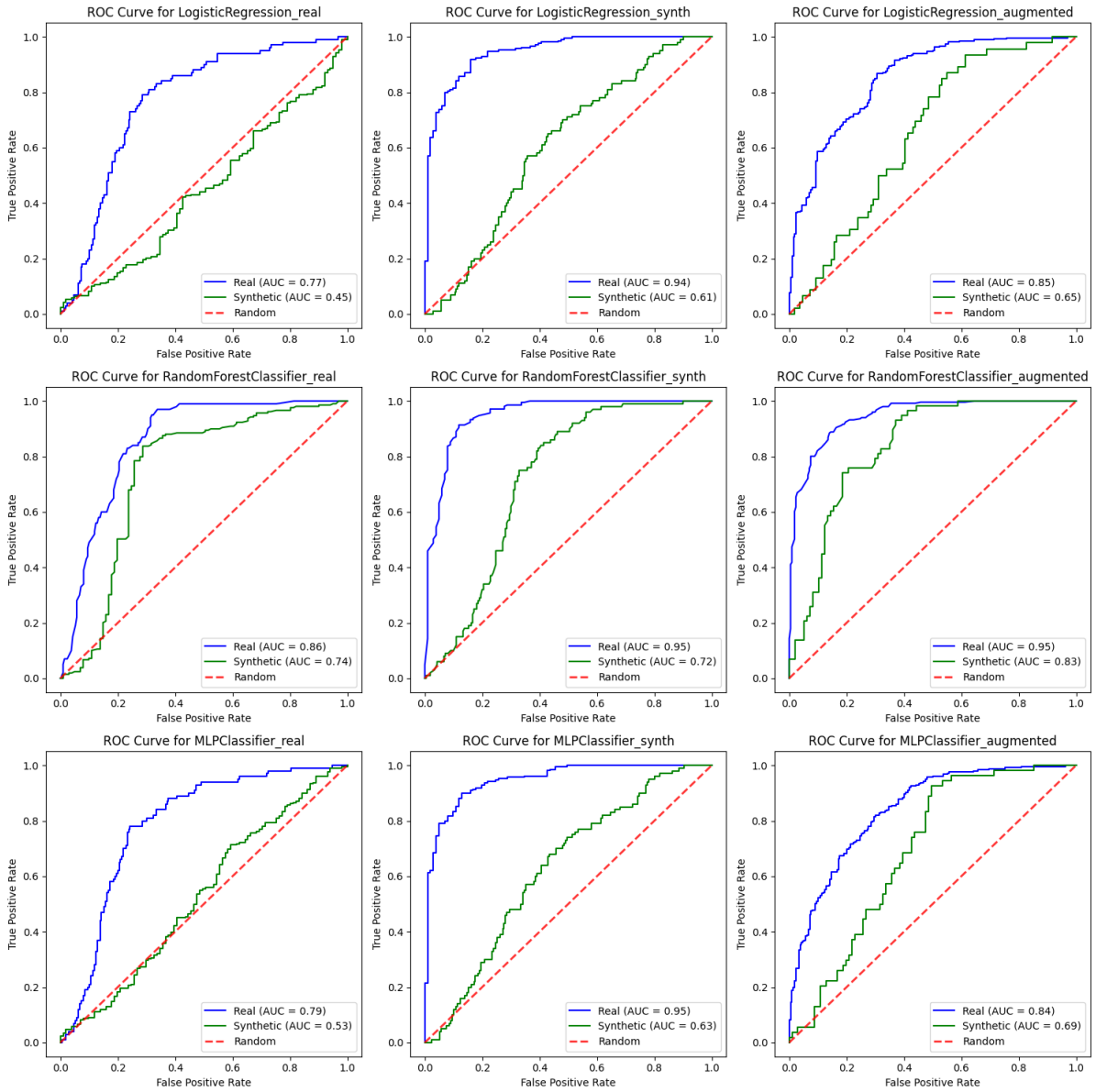


Figure A.7: All ROC curves for CopulaGAN's *Lower Back Pain* dataset.

GAN Model	Classifier	Train Data	F1 Real	F1 Synth/Augmented	Difference
CTGAN	Logreg	Real	0.6866	0.2812	0.4054
	Logreg	Synth	0.5239	0.8904	-0.3665
	Logreg	Augmented	0.6638	0.7973	-0.1335
	RFC	Real	0.7817	0.4188	0.3629
	RFC	Synth	0.5506	0.9201	-0.3695
	RFC	Augmented	0.7805	0.8543	-0.0738
	MLP	Real	0.7418	0.3589	0.3829
	MLP	Synth	0.5309	0.8955	-0.3646
	MLP	Augmented	0.7004	0.8104	-0.11
CopulaGAN	Logreg	Real	0.6866	0.1922	0.4944
	Logreg	Synth	0.2968	0.881	-0.5842
	Logreg	Augmented	0.5882	0.787	-0.1988
	RFC	Real	0.7807	0.2196	0.5611
	RFC	Synth	0.5692	0.9008	-0.3316
	RFC	Augmented	0.7455	0.8683	-0.1228
	MLP	Real	0.7381	0.2212	0.5169
	MLP	Synth	0.3697	0.8891	-0.5194
	MLP	Augmented	0.6409	0.7563	-0.1154

Table A.1: All F1 scores from CTGAN and CopulaGAN on the *Lower Back Pain* dataset.

Appendix B: Estimation of Obesity Levels

B.1 Basic Statistical Check

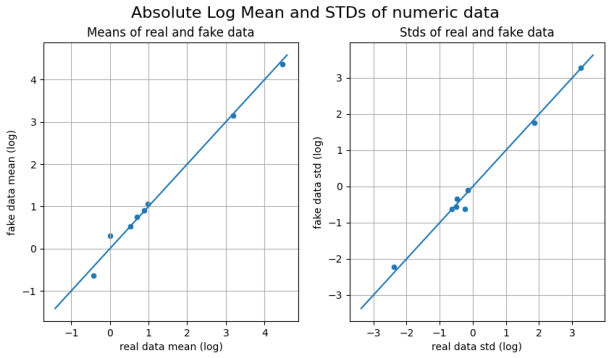


Figure B.8: Absolute Log Mean and STD of numeric data for Obesity dataset generated by CopulaGAN.

B.2 Correlations

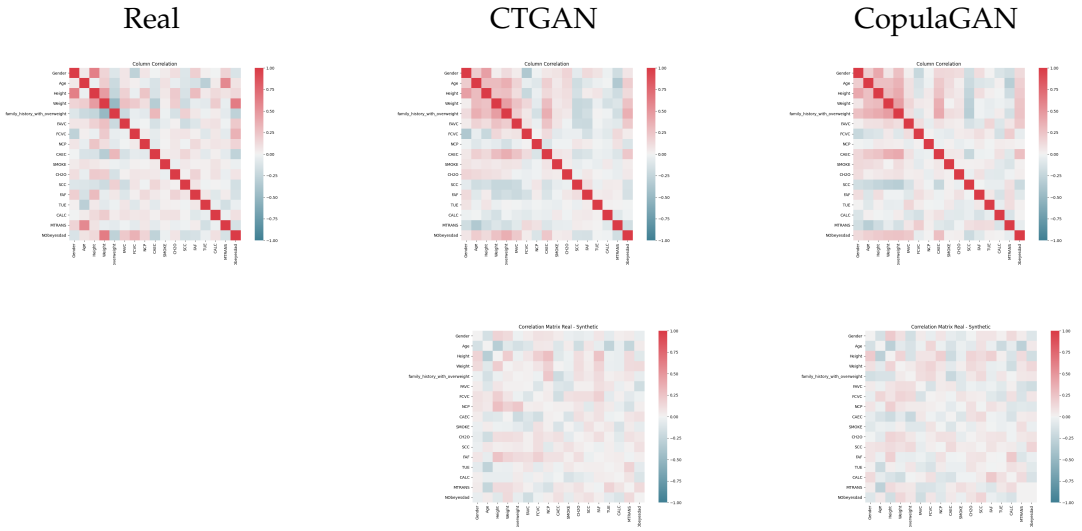
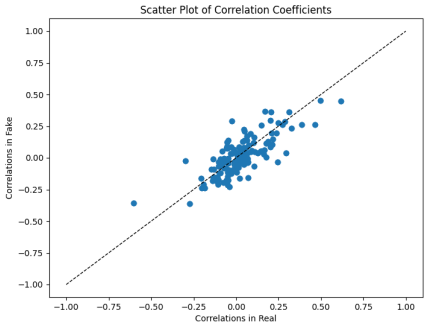


Figure B.9: Comparison of correlation matrices for the *Obesity* dataset (top row), along with the corresponding difference matrices (bottom row).

CTGAN



CopulaGAN

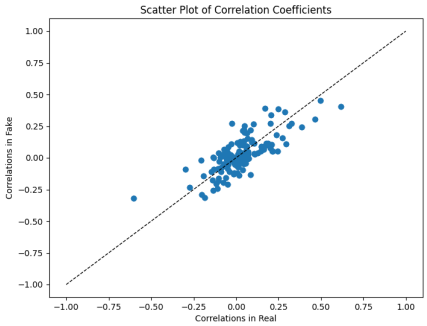


Figure B.10: Scatter plot for correlation coefficients on the *Obesity* dataset.

B.3 Column Distributions

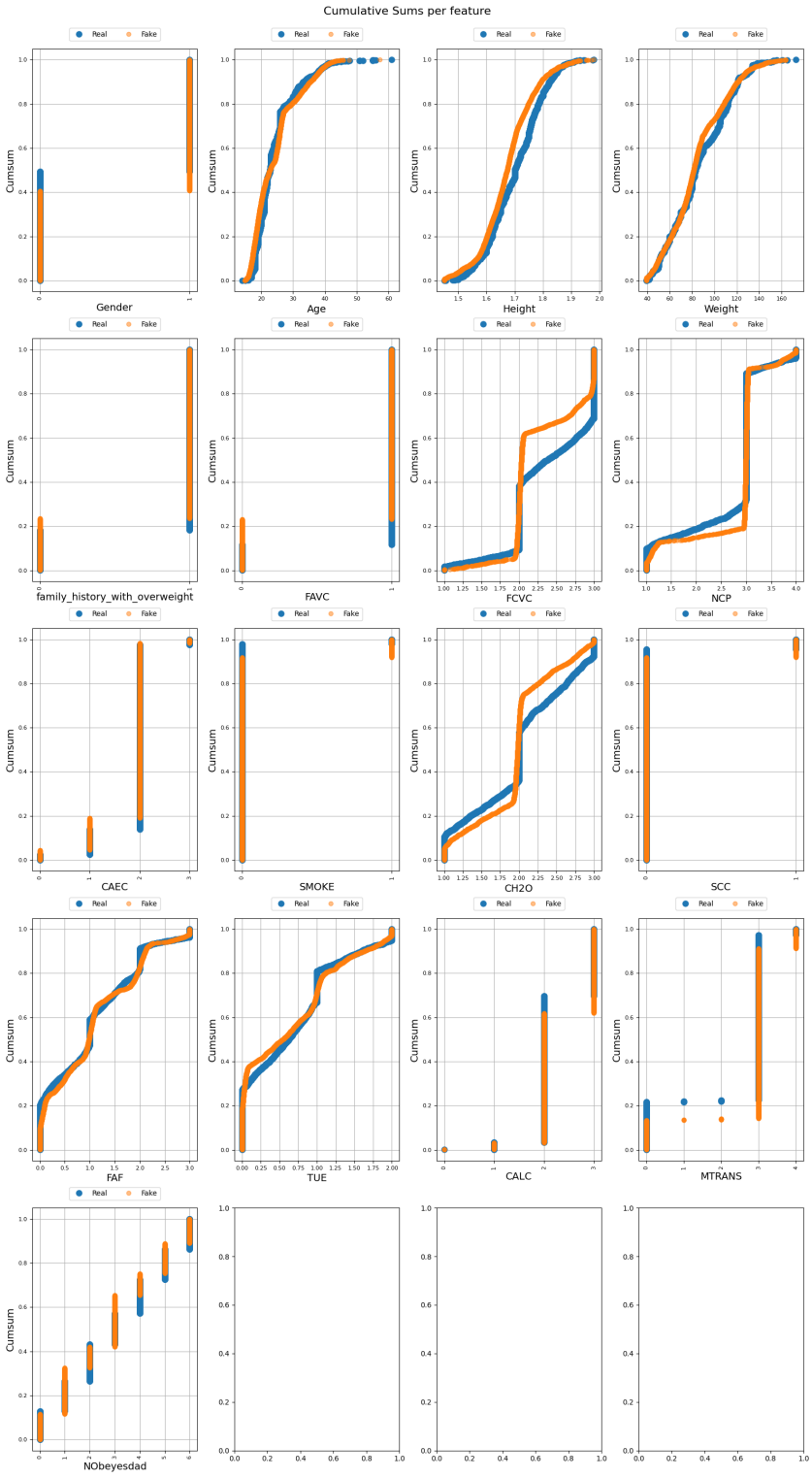


Figure B.11: Cumulative sums of each feature in the *Obesity* dataset for CTGAN.

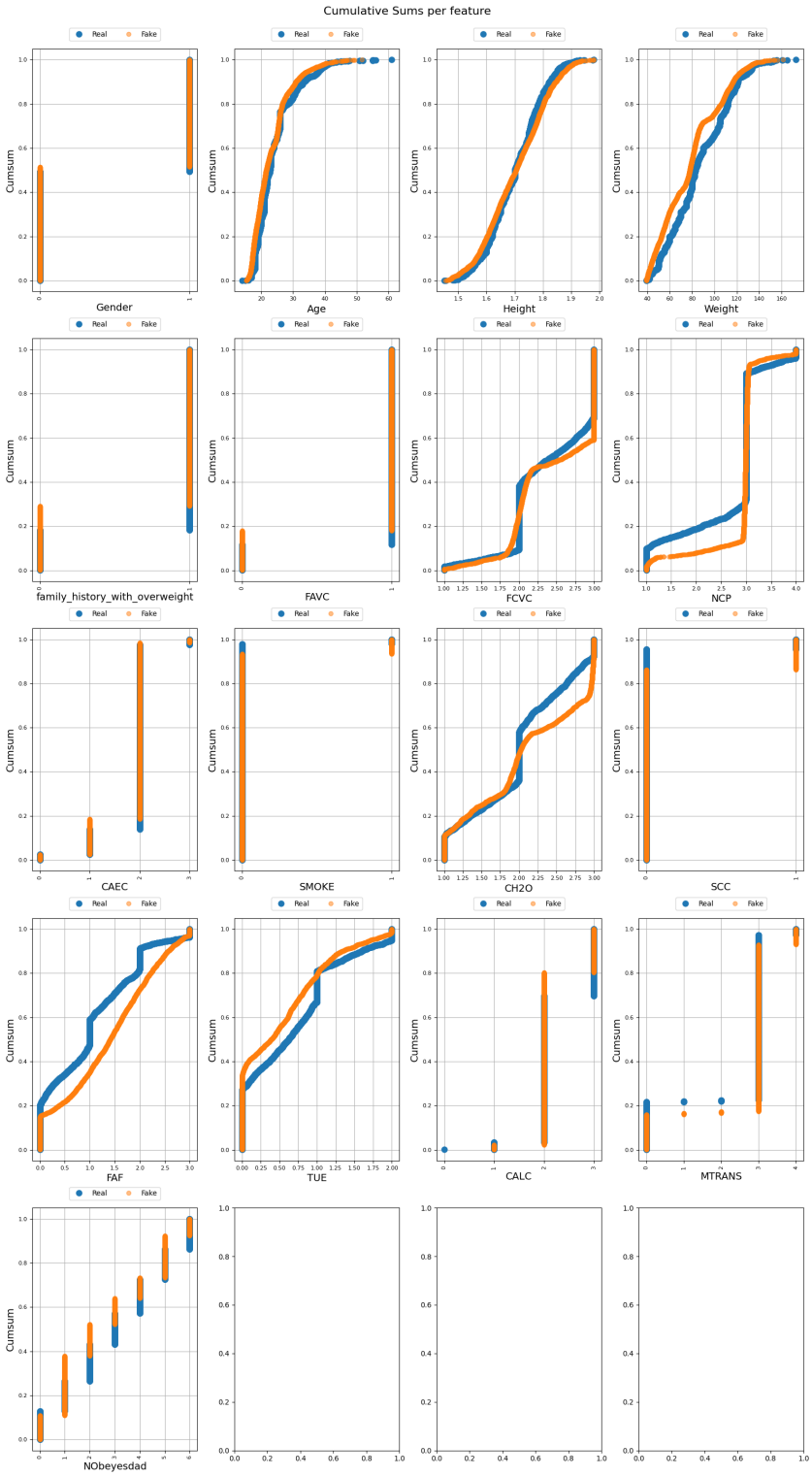


Figure B.12: Cumulative sums of each feature in the *Obesity* dataset for CopulaGAN.

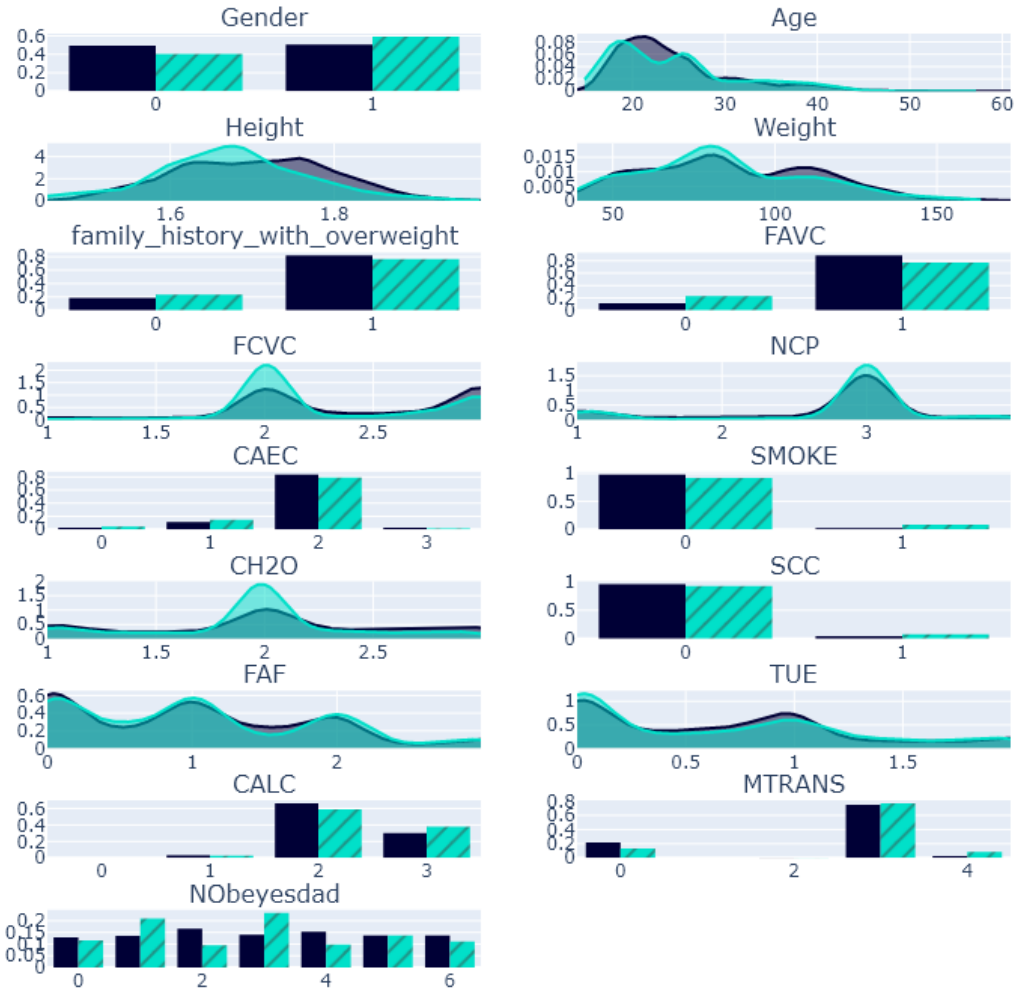


Figure B.13: Feature shape comparison of each column in the *Obesity* dataset for CTGAN

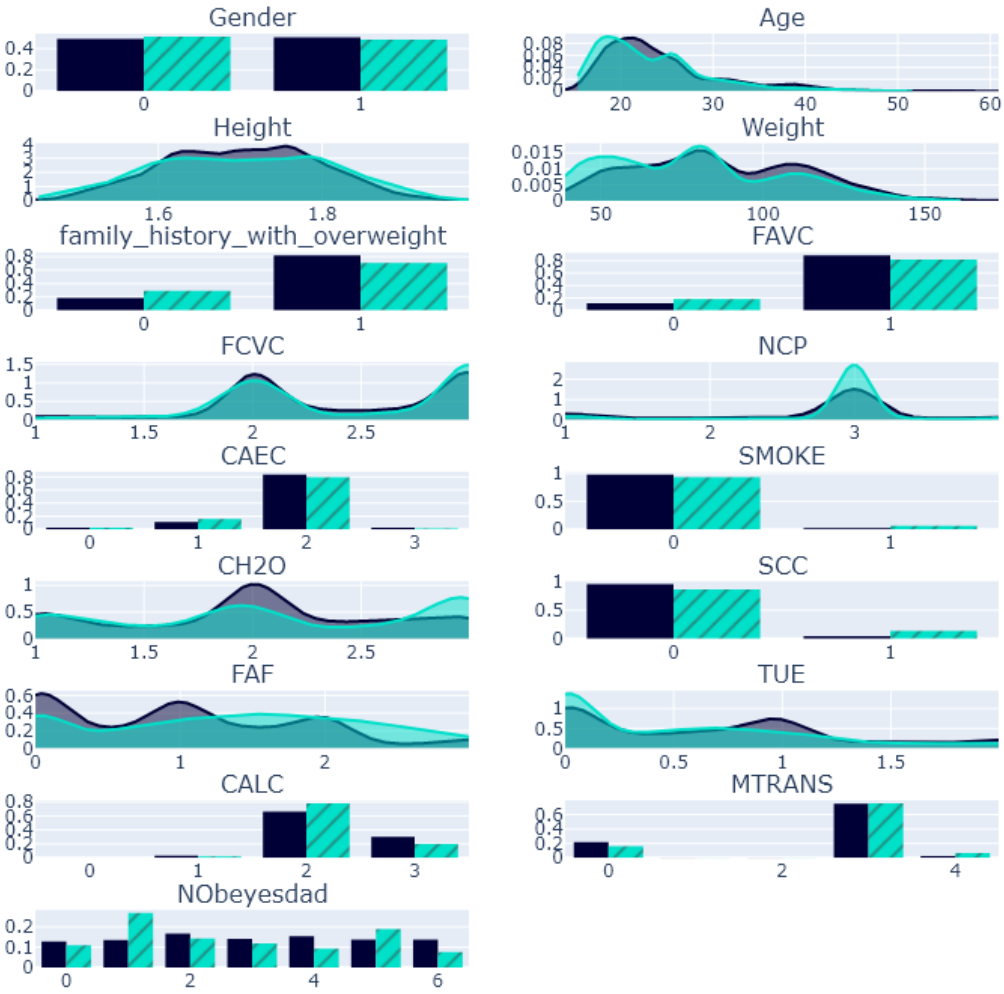


Figure B.14: Feature shape comparison of each column in the *Obesity* dataset for CopulaGAN

B.4 Classifier Evaluation

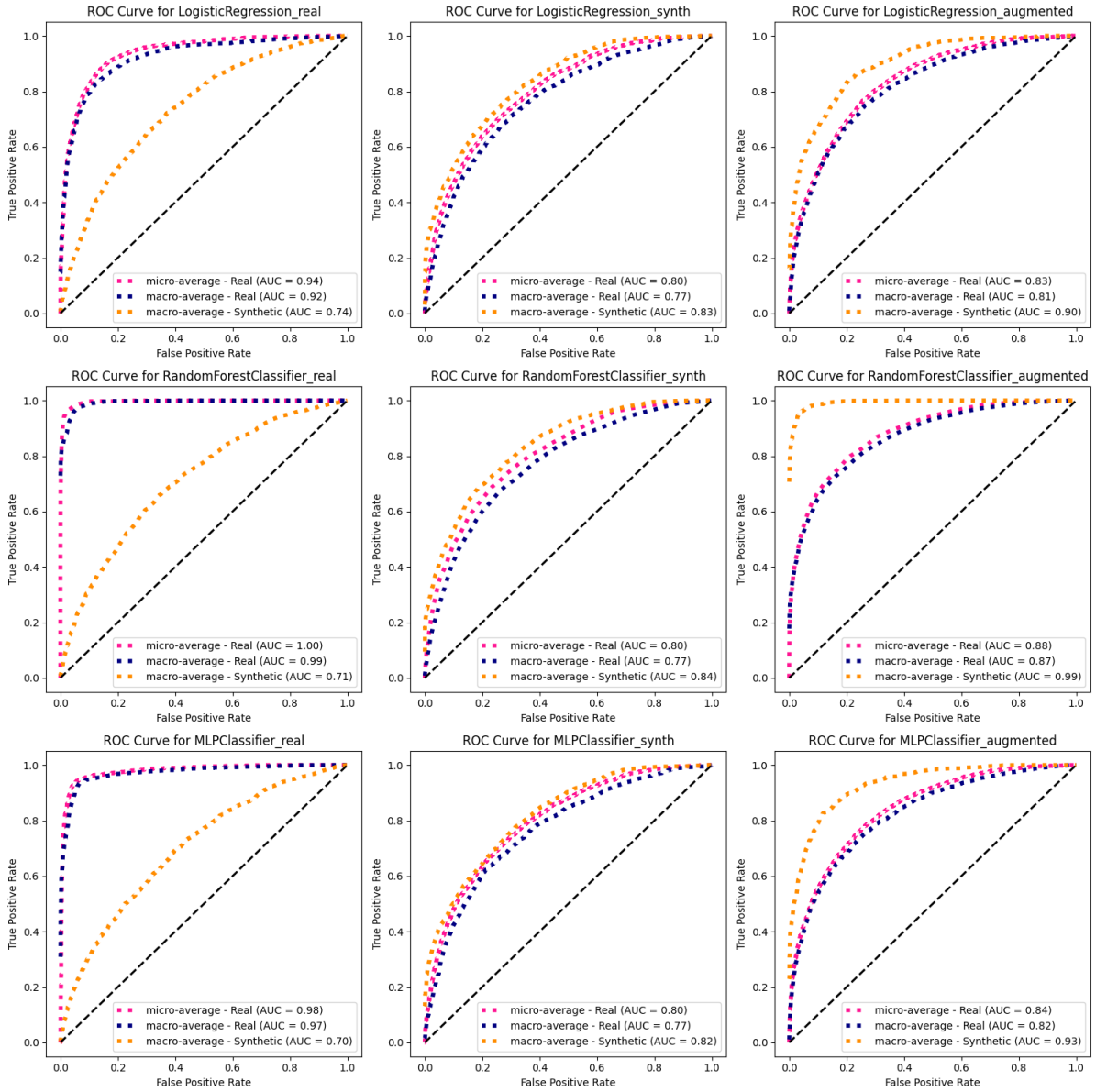


Figure B.15: All ROC curves for CTGAN's *Obesity* dataset.

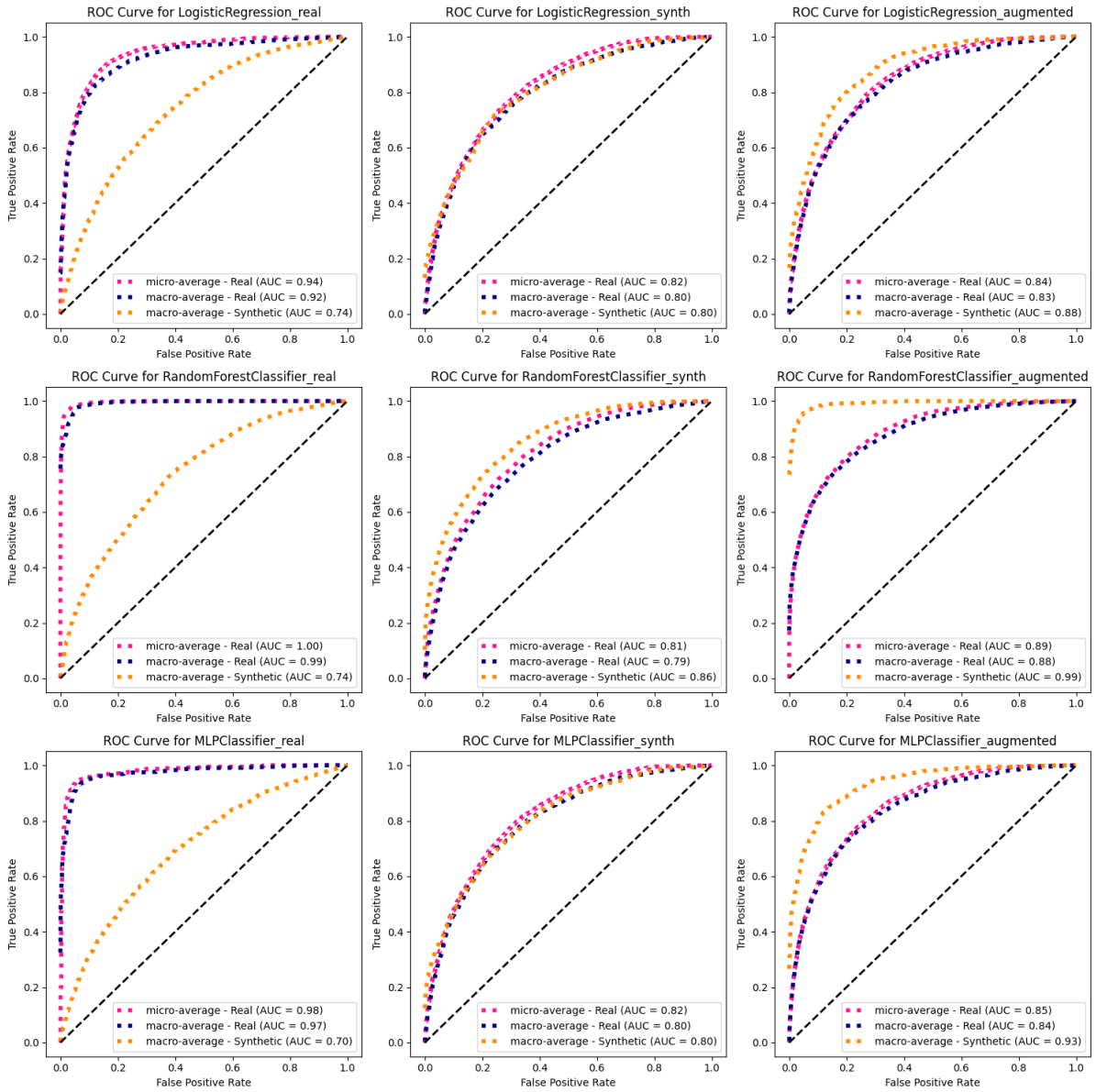


Figure B.16: All ROC curves for CopulaGAN's *Obesity* dataset.

GAN Model	Classifier	Train Data	F1 Real	F1 Synth/Augmented	Difference
CTGAN	Logreg	Real	0.6833	0.3481	0.3352
	Logreg	Synth	0.3963	0.3863	0.01
	Logreg	Augmented	0.5547	0.4431	0.1116
	RFC	Real	0.9356	0.3196	0.616
	RFC	Synth	0.3953	0.4144	-0.0191
	RFC	Augmented	0.9296	0.5871	0.3425
	MLP	Real	0.8609	0.3297	0.5312
	MLP	Synth	0.4205	0.3999	0.0206
	MLP	Augmented	0.5985	0.4879	0.1106
CopulaGAN	Logreg	Real	0.6833	0.3427	0.3406
	Logreg	Synth	0.4376	0.4156	0.022
	Logreg	Augmented	0.5211	0.4649	0.0562
	RFC	Real	0.9378	0.3149	0.6229
	RFC	Synth	0.4897	0.4251	0.0646
	RFC	Augmented	0.9207	0.5877	0.333
	MLP	Real	0.8696	0.3073	0.5623
	MLP	Synth	0.4451	0.42	0.0251
	MLP	Augmented	0.6703	0.5056	0.1647

Table B.2: All F1 scores from CTGAN and CopulaGAN on the *Obesity* dataset.

Appendix C: Cardiovascular Disease Prediction

C.1 Basic Statistical Check

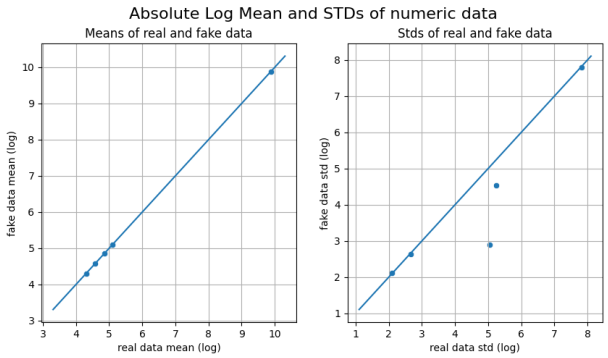
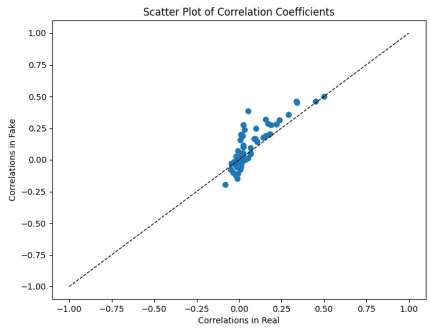


Figure C.17: Absolute Log Mean and STD of numeric data for Cardiovascular Disease dataset generated by CopulaGAN.

C.2 Correlations

CTGAN



CopulaGAN

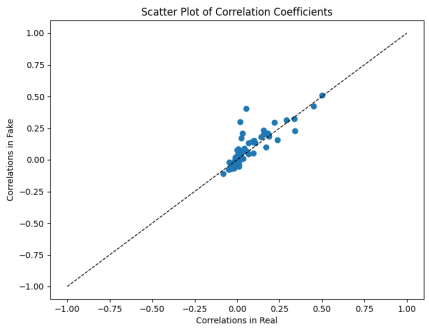


Figure C.18: Scatter plot for correlation coefficients on the *Cardiovascular Disease* dataset.

C.3 Column Distributions

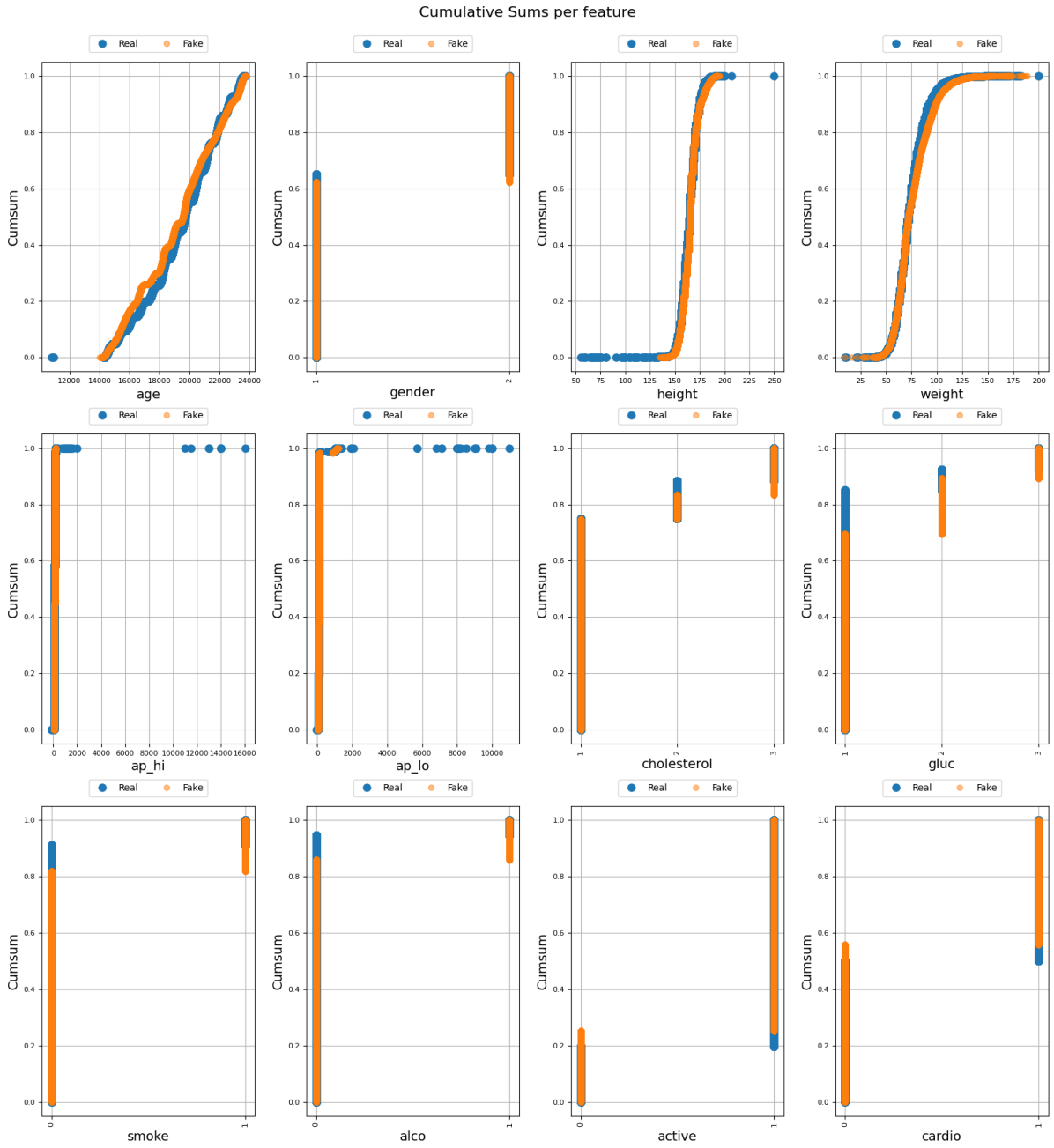


Figure C.19: Cumulative sums of each feature in the *Cardiovascular Disease* dataset for CTGAN.

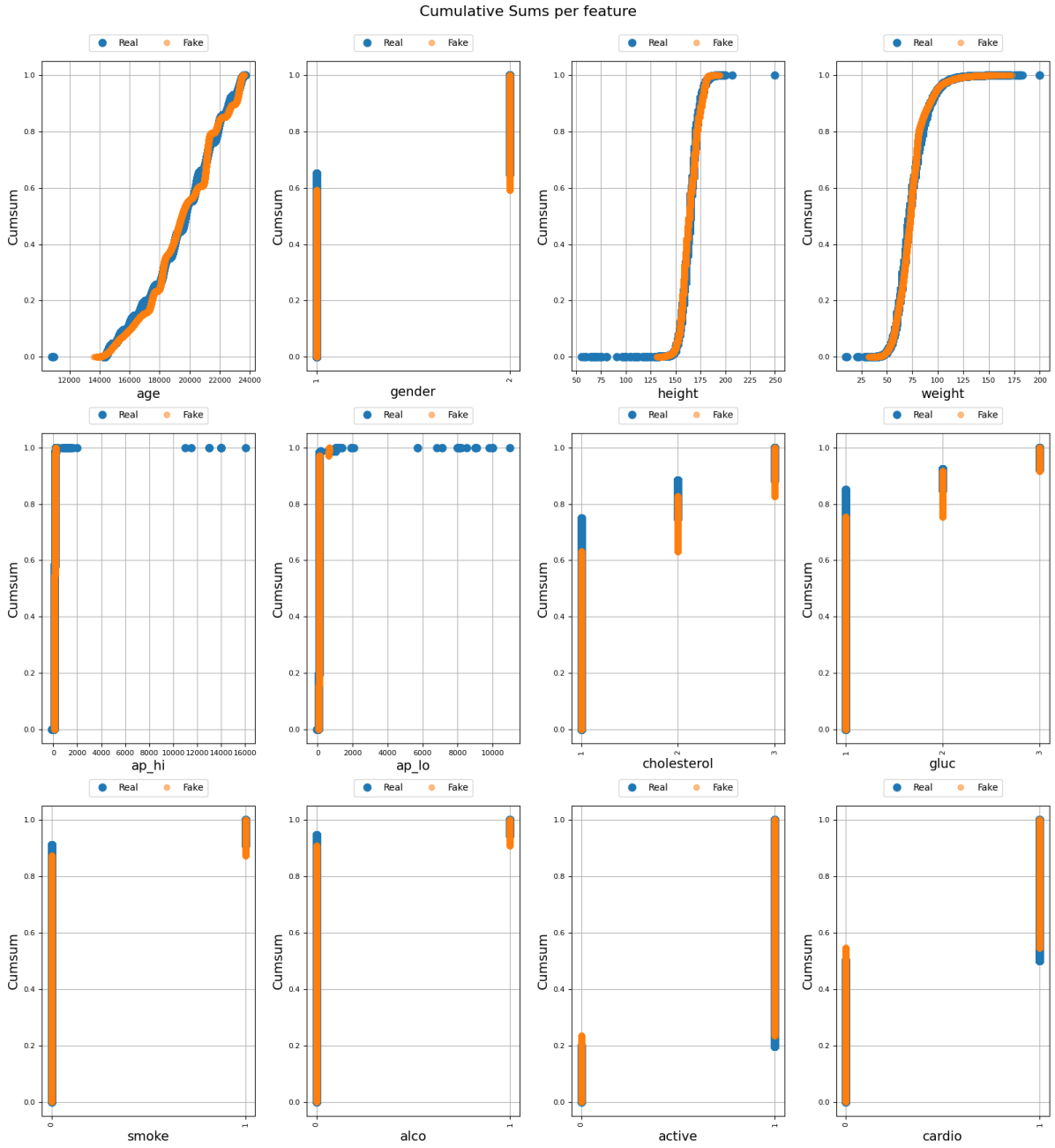


Figure C.20: Cumulative sums of each feature in the *Cardiovascular Disease* dataset for CopulaGAN.

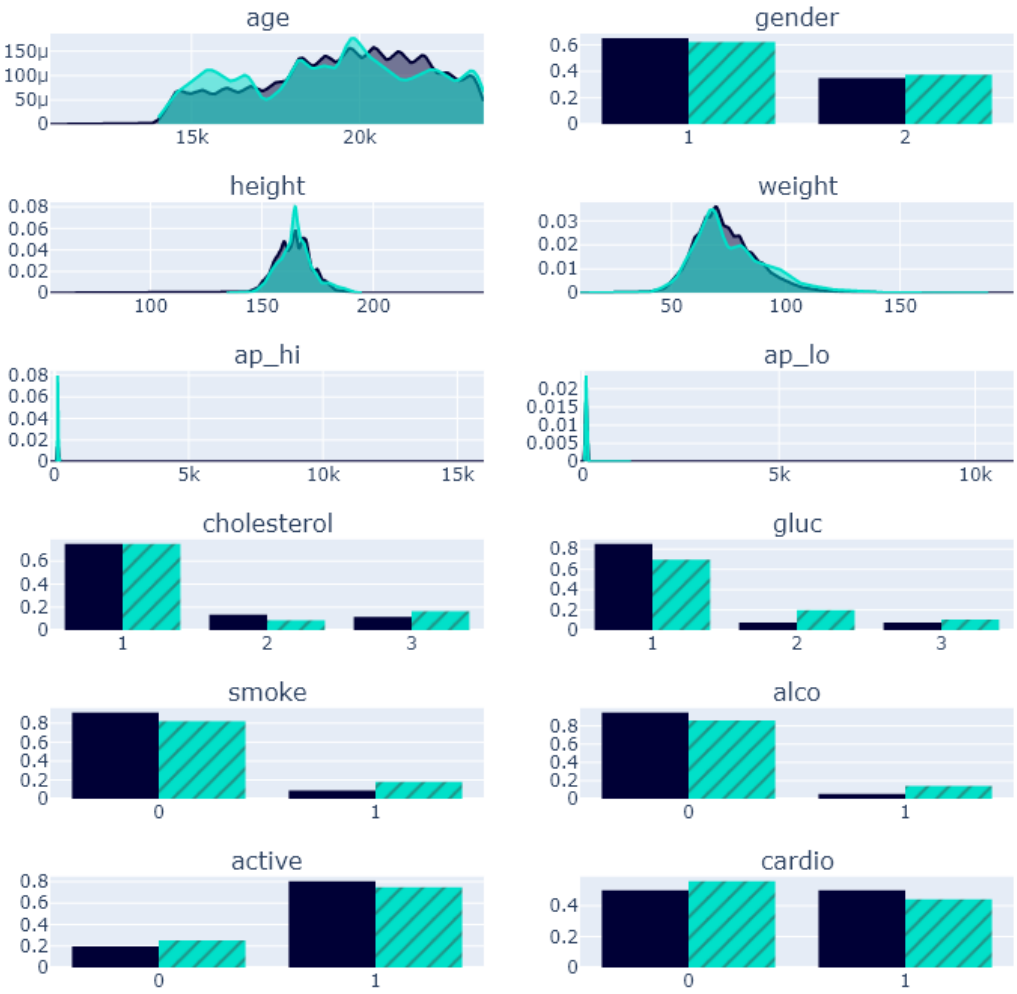


Figure C.21: Feature shape comparison of each column in the *Cardiovascular Disease* dataset for CTGAN.

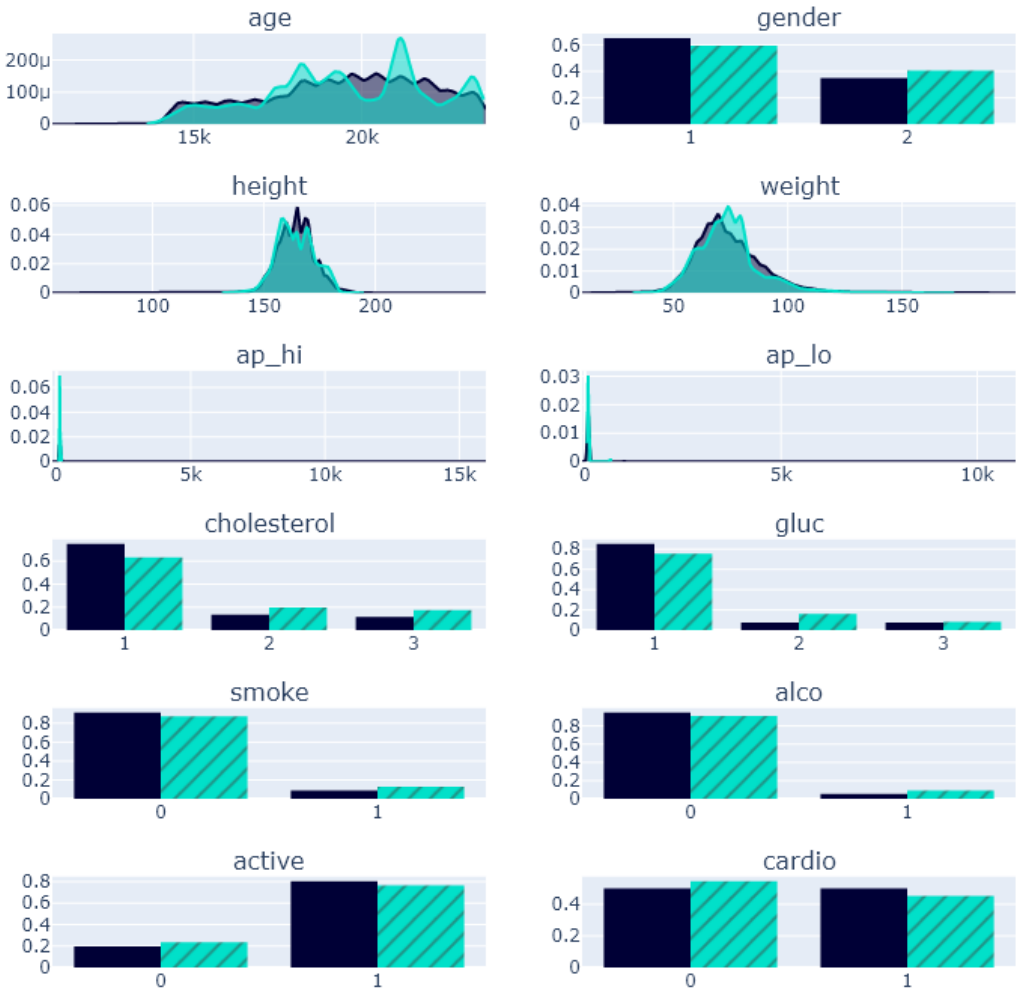


Figure C.22: Feature shape comparison of each column in the *Cardiovascular Disease* dataset for CopulaGAN.

C.4 Classifier Evaluation

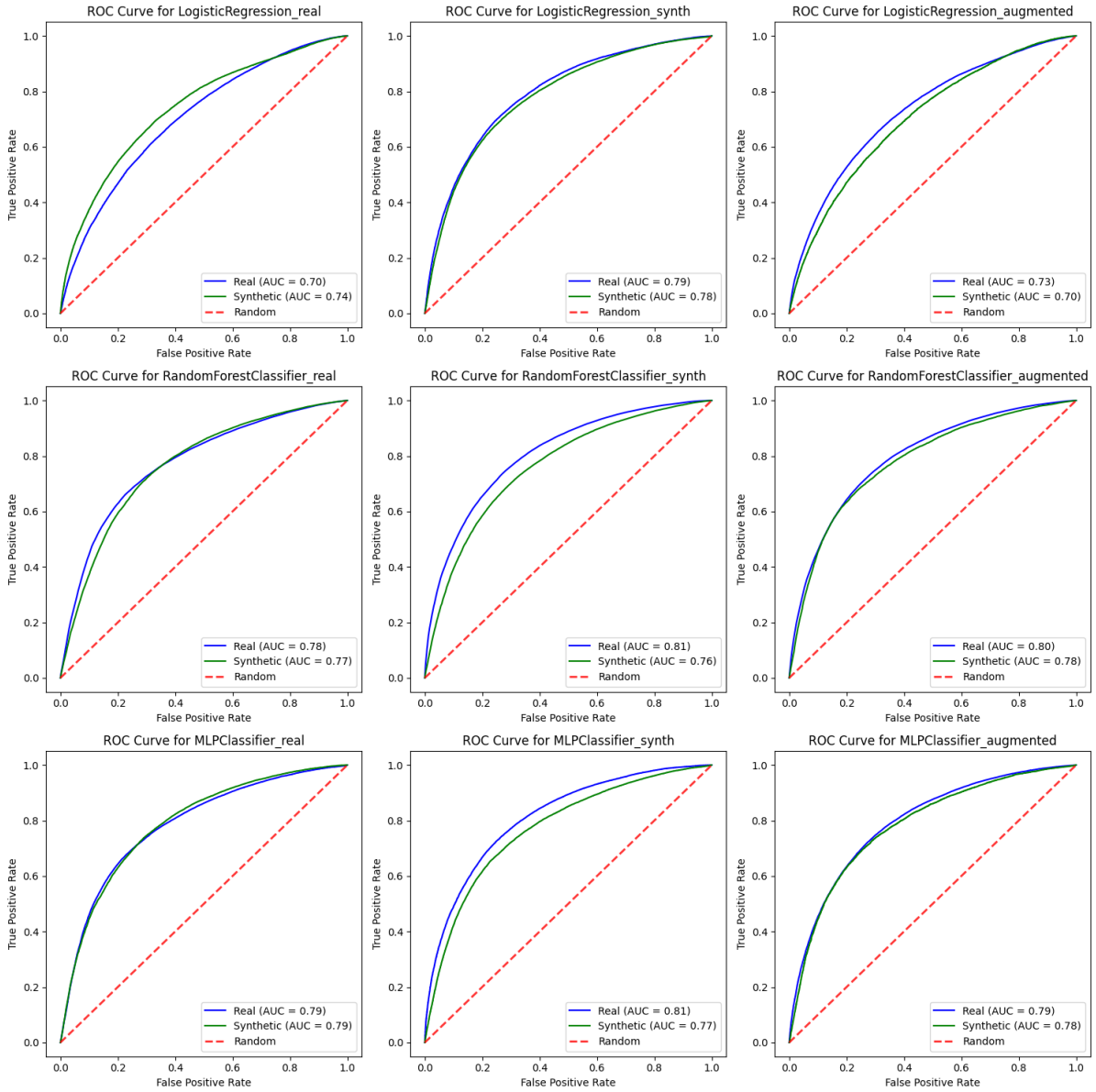


Figure C.23: All ROC curves for CTGAN's *Cardiovascular Disease* dataset.

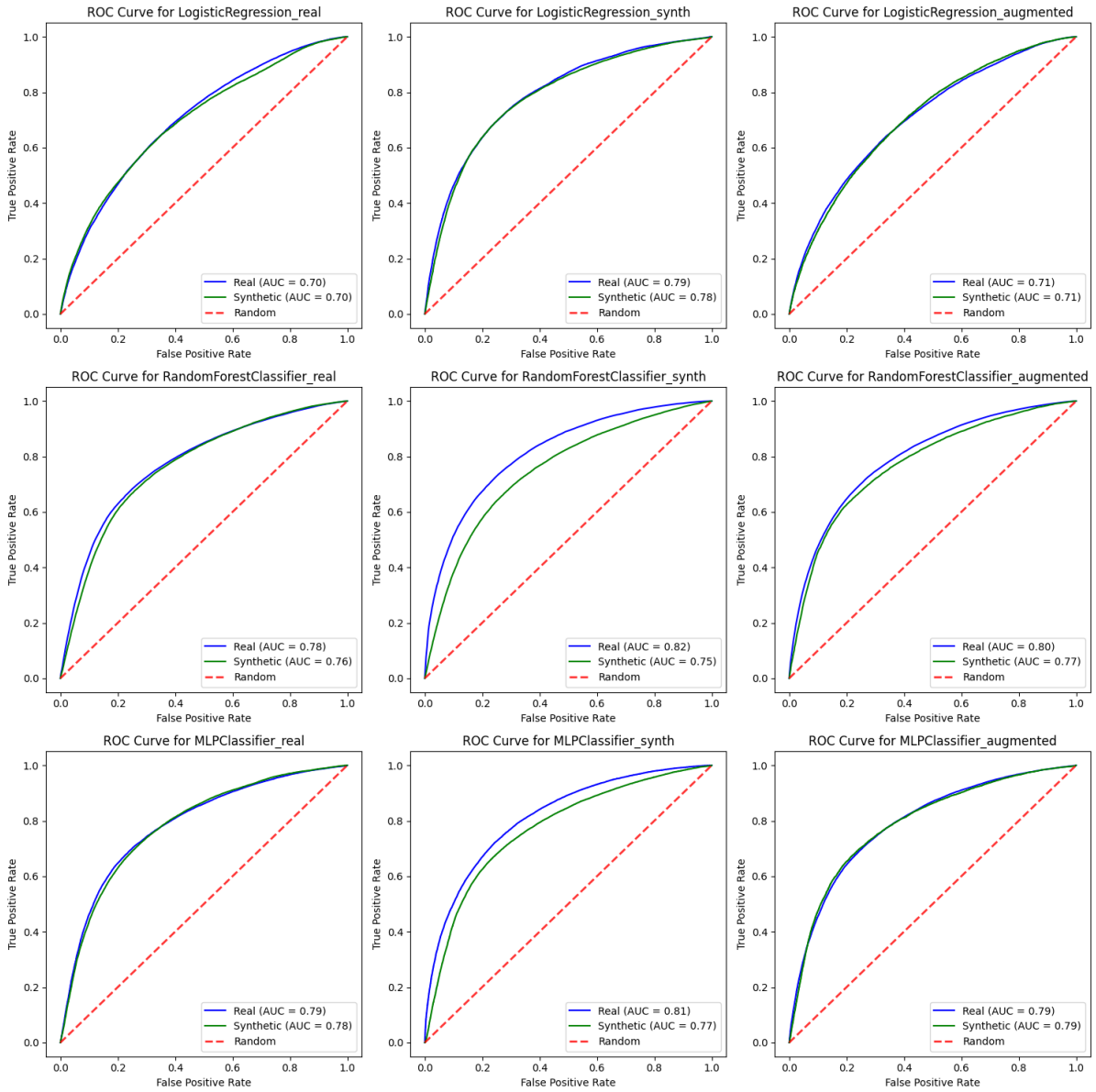


Figure C.24: All ROC curves for CopulaGAN's *Cardiovascular Disease* dataset.

GAN Model	Classifier	Train Data	F1 Real	F1 Synth/Augmented	Difference
CTGAN	Logreg	Real	0.6474	0.6837	-0.0363
	Logreg	Synth	0.6957	0.7245	-0.0288
	Logreg	Augmented	0.6406	0.6727	-0.0321
	RFC	Real	0.7152	0.7091	0.0061
	RFC	Synth	0.6906	0.7343	-0.0437
	RFC	Augmented	0.7183	0.7276	-0.0093
	MLP	Real	0.7224	0.7163	0.0061
	MLP	Synth	0.7021	0.7407	-0.0386
	MLP	Augmented	0.7165	0.725	-0.0085
CopulaGAN	Logreg	Real	0.6474	0.6422	0.0052
	Logreg	Synth	0.6893	0.7228	-0.0335
	Logreg	Augmented	0.6225	0.6482	-0.0257
	RFC	Real	0.7155	0.6981	0.0174
	RFC	Synth	0.6872	0.7422	-0.055
	RFC	Augmented	0.7122	0.7274	-0.0152
	MLP	Real	0.7215	0.6971	0.0244
	MLP	Synth	0.7048	0.7402	-0.0354
	MLP	Augmented	0.713	0.7233	-0.0103

Table C.3: All F1 scores from CTGAN and CopulaGAN on the *Cardiovascular Disease* dataset.