POLITECNICO DI TORINO

Master's degree in Computer Engineering

Master's Degree Thesis

Research and development of methods for the generation of synthetic data for the protection of privacy and the reduction of the risk of bias

Supervisors

Prof. Antonio Vetro'

Ing. Marco Rondina

Candidate

Giacomo Fantino

Graduation Session July 2024

a.a. 2023/2024

Abstract

The rapid advancement of artificial intelligence (AI) in recent years has been nothing short of remarkable. AI systems have demonstrated remarkable capabilities in a wide range of domains, from natural language processing and computer vision to classification and decision making. This growth has been fuelled by the availability of vast amounts of digital data, coupled with significant improvements in computing power and algorithmic techniques. The amount of data required continues to grow, and this data hunger raises new issues.

Oversampling is a technique based on obtaining new data from a trusted and well-documented dataset to improve a model's performance. However, historical biases embedded in training data can lead to AI systems perpetuating and amplifying societal biases, with serious implications for fairness and equity [3]. Finally, as AI systems become more pervasive in our daily lives, they collect and process vast amounts of personal data. The presence of this kind of data poses some limitations in terms of sharing and training models with this data, hence the emergence of a vast literature on data privacy.

The aim of this work of thesis was to study and compare different methods for the generation of synthetic data, evaluating their impact while trying to solve the above scenarios. After reviewing traditional methods such as Gaussian Mixture Models (GMMs) and SMOTE, I studied the field of mechanisms based on Neural Networks, with a focus on Generative Adversarial Networks [16].

While GANs were originally developed for image generation, several approaches have been developed specifically for tabular data. CTGAN [31] and CTAB-GAN+ [35] are the most recent and promising and have demonstrated the capabilities of GANs for tabular data. Their strength lies in their better ability to model distributions of categorical data, while learning the complex relationships typically found in tabular data.

I started by experimenting with the **oversampling** scenario. GANs have been tested on specific datasets, with the presence of categorical data and skewed continuous data. I have worked with the HTRU2 dataset [22], known to be an imbalanced dataset with only continuous features, so I was able to compare the above mechanisms with GMM and SMOTE. My experiment was based on reducing the proportion of minority class samples from the dataset, generating

the synthetic data and measuring the improvement in performance with and without synthetic data. What I found was that SMOTE had a weaker impact as the percentage of the minority class decreased, while GMM on average had the best improvement. Regarding the GANs, both can compete with the GMM, with CTGAN occasionally giving the best improvement. This highlights a need for improvement for GANs: due to the resources, both in terms of hardware and time required, it is still worth using a GMM for these specific datasets.

Next step I evaluated a **privacy** scenario: when a dataset contains sensitive information, researchers have problems at sharing such data because of the possible consequences. The use of synthetic data has emerged in this area, with the aim of providing the possibility of sharing data with similar statistical information to the original, with the possibility of respecting privacy. However, this technique has been undermined by several attacks where an attacker can infer information about the original data from the synthetic data. I have evaluated synthetic data from different generators against a specific attack: the Membership Inference Attack (MIA) [7], where an attacker can infer whether a particular sample belongs to the training data used to compute the synthetic data. The results show how GAN-based generators can withstand this attack, while a perturbation method fails to generate robust synthetic data from the same training data. GANs also proved to be more robust: as the size of the training data decreased, the accuracy of the attack did not change. An explanation of these performances can be obtained by using the Distance to Closest Record (DCR) and the Nearest Neighbour Distance Ratio (NNDR) [35], which are based on measuring for each synthetic sample the minimum distance between that sample and all the samples from the training data. These show that the synthetic data produced by both GANs have a higher average value compared to the synthetic data produced by the perturbation methods, indicating that they were able to generalize better.

Finally, I evaluated the impact of synthetic data generated by CTAB-GAN+ in a **fairness** scenario, where we want to introduce synthetic data in the training set to modify the decision boundaries of a model due to a bias in the decisions. For this part, I used two different approaches to study the decision of a model: Explainable AI (XAI) and fairness measures [6]. Explainable AI explores a range of methods to assess how a model works internally and to explain the predictions, while fairness measures use a range of criteria to assess whether a model discriminates against a minority group. For the XAI part, I showed that by including synthetic data, the importance of sensitive features in the decision processes is reduced, resulting in a model that is less prone to making a direct discriminatory decision. Similar results were obtained when looking at fairness measures: the difference between groups (with respect to a sensitive feature) tends to decrease, resulting in a less discriminative model. There were still some differences that were not captured by the XAI approach, due to some historical biases in the training data that were still present in the synthetic data. This is a limitation of this approach: since the data set must capture biases to be a good reflection of our society, these will be built into the model. A model that does not discriminate is a model without bias, but the only possibility is that the training data does not reflect the reality it needs to capture.

Overall, these results provide a positive direction for the generative mechanism of synthetic data based on a neural architecture, taking into account the current limitations:

- For an oversampling application, GANs provide state-of-the-art results with categorical and mixed-type data. The new challenge is to create a new methodology that can work with a wider variety of datasets.
- The synthetic data produced with GAN-based generators can withstand a membership inference attack, so the use of this data ensures a good level of privacy.
- Bearing in mind that it's a technical solution to a socio-technical problem, including synthetic data in a training set can help reduce the bias of a model.

Although this thesis has some limitations due to the lack of a variety of datasets for each experiment and generative methods, the results are promising and provide a starting point for research and development of new state-of-the-art methods for generating synthetic data with an acceptable trade-off between privacy preservation, fairness and accuracy.

Ai miei genitori per tutto quello che hanno fatto per me,
Ai miei amici per tutte le esperienze fatte insieme,
A quei professori che mi hanno insegnato l'arte di condividere la conoscenza.

Contents

Chapter 1

Introduction

The rapid advancement of artificial intelligence (AI) in recent years has been nothing short of remarkable. AI systems have demonstrated remarkable capabilities in a wide range of domains, from natural language processing and computer vision to classification and decision making. This growth has been fueled by the availability of vast amounts of digital data, coupled with significant improvements in computing power and algorithmic techniques.

However, as AI systems become increasingly sophisticated and ubiquitous, concerns have emerged regarding their limitations and potential pitfalls. A key challenge is the AI's reliance on data, which can often be biased, incomplete, or unrepresentative of the full diversity of human society. The overall quality of the outcomes of AI depends heavily on the data [14] and historical biases embedded in training data can lead to AI systems perpetuating and amplifying societal prejudices, with serious implications for fairness and equity.

Another pressing issue is the impact of AI on individual privacy. As AI systems become more pervasive in our daily lives, they collect and process vast amounts of personal data, raising concerns about the protection of sensitive information and the potential for misuse or abuse. The presence of this kind of data poses some limitations in terms of sharing and training models with this data, hence a vast literature for anonymization of data was born.

1.1 What is AI?

Thoughts on the general idea of a 'thinking machine' can be traced back many centuries, for example in the works of Gottfried Wilhelm von Leibniz and Ada

Lovelace. Nonetheless the terms actually originated in the 1950s by John McCarthy, that later in 1956 organized the Dartmouth workshop [24]. Today it's widely considered to be the founding event of artificial intelligence as a field.

In the 1980s the field emerged with the introduction of Machine Learning, where the computer improves its own ability to carry out tasks by analysing new data, without a human needing to give instructions in the form of a program[1]. This new technology brought better performances and the possibility of applying AI to new and more complex task. With the improvement of computer architectures, the adoption of new computing devices, like Graphic Processing Unit (GPU), and the introduction of neural networks a new industry was created. An increasing number of commercial applications are nowadays based on AI systems, sometimes without really understanding how AI can impact our society.

1.2 What is Data?

Today, most artificial intelligence is based on data pattern recognition [5]: AI models learn an internal representation of data patterns. This internal representation is then used to perform a given task on other instances of data that have not yet been encountered. This fundamental principle at the heart of all AI software introduces the basic raw materials needed to train the algorithms: data.

The amount of data needed is always bigger and this data hunger raises new issues. Datasets are not simply operational instruments of digital knowledge production but one of the core elements of AI. As previously mentioned, AI looks at the data and incorporate any pattern useful for the task, without questioning its impact. For this reason the data should be:

- Diverse and varied.
- Unbiased.
- High-quality.
- Comprehensive.

In most cases, except for unsupervised learning, data should be annotated in order to train a model to perform a classification task. Annotating data means

[1]Definition from the Cambridge Advanced Learner's Dictionary & Thesaurus

manually classifying it, providing a label that the model can consider as ground truth. A desired property of labelling is objectivity: however, the subjectivity of human annotation makes this a very ambitious goal [9].

New techniques have been proposed to address the above problems, including oversampling: starting from a trusted and well documented dataset, include new synthetic data generated based on the original data, with the objective of enhancing it and improving its usefulness for training a Machine Learning model.

1.3 The impact of AI

One of the key book for understanding the impact of AI in our world is *Atlas of AI* by Kate Crawford [9]. Earth's resources are one fundamental element of the AI supply chain. Furthermore, the energy required to train model is significant: the carbon footprint of training a single big language model is equal to around 300,000 kg of carbon dioxide emissions [11].

In this thesis one particular impact is taken into consideration: what is the impact of a data-drive automated decision makers for specific group of people? Considering that datasets always carry the historical biases of our society, can we keep using our models while also minimizing the impact of such models?

For this particular problem many new field of research started in the past few years: Explainable AI (XAI) wants to study how the models internally make decisions, while fairness measures [3] try to evaluate the unfairness of a model via different metrics. I will apply this techniques to assets the impact of synthetic data.

Chapter 2

Synthetic Data

2.1 Background

Synthetic data are obtained by a generative process based on properties of real data. Over the years, this technique has become known and studied because it allows certain objectives on the data to be guaranteed:

- Data augmentation
- Anonymization.
- Semi-supervised learning: Using labeled and unlabeled data to train neural networks
- Balancing unbalanced learning contexts.

Finding an appropriate data generation technique depends on several factors, including the domain of the data and its type. Other factors are:

- The importance of statistical information of real data and the relationship between features.
- The task of the model.
- The role of synthetic data in the process (anonymization, balancing, augmentation...).

Let's look at an example of this: if our goal is to balance a minority class we mainly want to add new samples without giving too much weight to the relationships between features. On the other hand, if we want to anonymize the dataset, we need to ensure that we have preserved the original statistical information (since we are only going to use the synthetic dataset). It will also be critical to study and define metrics to compare the original dataset with the new one to verify that we have met the goals we set for ourselves.

More formally, if we define our initial dataset D, defined by samples and labels:

$$D = (x_i, y_i), i = 1..l \tag{2.1}$$

We can then define a generator (in this case modeled with a function) and the synthetic dataset D^s as:

$$\begin{aligned} D^s &= (x^s, l) \\ x^s &= f_{gen}(x, \tau) \end{aligned} \tag{2.2}$$

Where l is the label of the original data x. τ incorporates the hyperparameters of the function, which are called **generation policy**.

We can already define two metrics:

- Similarity: the more similar D and D^s are (from a statistical point of view) the better.
- Performance: the performance of the model trained by D^s is better or equal to the model trained with D (in case of class re-balancing the model is trained using $D \cup D^s$).

In general what we expect from the f_{gen} function is that D^s should have a data distribution similar to D without going to duplicate data:

$$\mathbb{P} \approx \mathbb{P}^s \wedge x_i \neq x_i \forall x_i \in D, x_j \in D^s \tag{2.3}$$

2.2 Evaluation dimensions

We will now go on to see an overview of different applications, with emphasis on the properties they require of our generative models and then going on to understand the caveats that are used.

2.2.1 Privacy

The privacy issue arises when there is sensitive information in our dataset that can be used to identify an individual. As a result, the use of these datasets in applications such as machine learning and data mining is very limited. Hence the idea of using the synthetic data to create an anonymized version of the original dataset. The aim is to make the original subjects unidentifiable without without compromising performances on the task.

To check the privacy of a dataset we use **differential privacy (DP)** [18]. In the context of synthetic data it translates to using two privacy parameters: ϵ and δ, where given a dataset D for all element x in D:

$$\begin{aligned} \mathbb{P}(f_{gen}(D) \in S) \leq e^{\epsilon} \cdot \mathbb{P}(f_{gen}(D/x) \in S) + \delta \\ \forall S \in Range(f_{gen}) \end{aligned} \tag{2.4}$$

This property became very popular because it is not affected by post-processing: any ML pipeline will maintain the same level of (ϵ, δ)-differential privacy. A more in depth analysis of DP will be given later.

2.2.2 Regularization

Many models rely on the assumptions that the training data is clean, the labels are correctly assigned, and the source of the data is fixed. When one or more of these assumptions are not met, model performance can deteriorate. Synthetic data can be used to counteract:

- Overfitting.
- Training set too small.
- High dimensionality.
- Outlier.
- Wrong labels.

A well-known technique in the field of computer vision and NLP is **data augmentation**, where for each item in a random subset of data is used to create a slightly modified copy of it (in the case of CV by changing the coloring and adding blur) and then add it into the training set.

A more general technique is called **Mixup** [33]: taken two random samples (x_i, y_i) and (x_j, y_j) we generate:

$$\begin{aligned} x^s &= \lambda x_i + (1 - \lambda)x_j \\ y^s &= \lambda y_i + (1 - \lambda)y_j \\ \lambda &= Beta(\alpha, \alpha) \end{aligned} \tag{2.5}$$

2.2.3 Oversampling

Another assumption employed by many models is that the frequency of classes in the training set is similar. Datasets skewed toward one class tend to perform worse. Oversampling is a special case of regularization, where we are interested in generating multiple samples of a set of minority classes. The idea is to generate D^s such that $D \cup D^s$ is balanced.

2.2.4 Fairness

Traditional AI models are often trained on real-world datasets, which can inadvertently reflect societal biases and inequalities. These biases can then be propagated and amplified when the models are deployed. Synthetic data offers a solution, as it can be carefully curated to ensure fair representation of diverse populations and to eliminate unwanted biases. By training AI models on datasets enriched with synthetic data that capture the desired characteristics and distributions, we can create AI systems that are fairer and more inclusive, without the limitations and biases inherent in real-world data.

2.3 Data quality assessment

The most obvious technique is to train a new classifier and measure performance. If they have improved our synthetic data are working, otherwise we have more work to do. Since the training phase of a model is in many cases time-consuming (in terms of time and resources) the question arises whether there are not other ways to evaluate synthetic data without having to train a new model.

Kullback-Leibler (KL) divergence is a metric often used to evaluate generative models [28]. It is defined as follows:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \cdot \log \frac{P(x)}{Q(x)} \tag{2.6}$$

Where P and Q are the distributions and X a set of values. It can easily be seen that if P and Q are very similar the logarithm will have a value close to zero and thus the summation will be almost zero. The problem with this metric is its difficult interpretation and the fact that it does not scale well on data with many dimensionalities.

We can then introduce the Jensen-Shannon (JS) divergence [28]:

$$\begin{aligned} D_{JS}(P||Q) = \frac{D_{KL}(P||M) + D_{KL}(Q||M)}{2} \\ M = \frac{P+Q}{2} \end{aligned} \tag{2.7}$$

Another metric is the propensity score [29]: a classifier (usually logistic regression) is trained using both original and synthetic data and using the data source (original or synthetic) as the target. The goal is to obtain the probability that a sample is synthetic. An example is the propensity mean squared error (pMSE):

$$pMSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{p}_i - 0.5)^2 \tag{2.8}$$

Where N is the set of original and synthetic data and $\hat{p}_i$ propensity score of the i-th sample (trivially is equal to 1 if the data is expected to be synthetic and 0 if real). If we get a value close to 0 then the classifier cannot distinguish between real and synthetic data and thus we have achieved the goal.

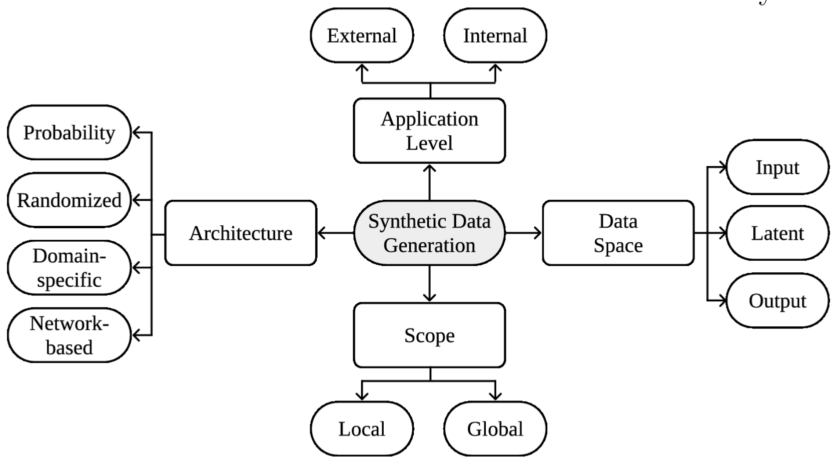
Lastly, difference in pair-wise correlation is adopted [35]: in order to assess how well feature interactions are preserved in the synthetic datasets, the pair-wise correlation matrix for the columns within the real and synthetic datasets individually. The Pearson correlation coefficient is used to measure the correlation between any two continuous features. Similarly, to measure the correlation between any two categorical features, the Theil uncertainty coefficient is used. Finally, the norm of the difference between the pair-wise correlation matrices for the real and synthetic datasets is used as measure.

Chapter 3

Generation mechanism

3.1 Taxonomy

From [13] I have adopted the following taxonomy to study and classify the different methods that have been created and used in recent years:



Where the following properties were used to characterize the different techniques:

- **Architecture**: analyze how internally the function f_{gen} generates the new data from D:
  - Probability: whether the function extracts data by sampling from a probabilistic function (computed from D).
  - Randomized: the original data are modified by random processes to obtain the new data.

  - Domain specific: data transformations are done using knowledge of the data structure (known **a priori** from the application domain).
  - Network-based: they use architectures based on neural networks.

- **Application level**: looking at the whole ML pipeline, where are we going to include the generative process?
  - Internal: alongside the primary the task.
  - External: used before the development of the ML pipeline.
- **Scope**: considering the properties of the original dataset:
  - Global: whether trying to maintain the distribution or statistical properties of D.
  - Local: is generated by considering a subset or a single element of D, generating new data that is more accurate but partly losing the distribution.
- **Data space**: refers to what type of data the transformation is applied to, whether to raw data (Input), an embedded representation (Latent) or on target feature (Output).

Since almost all generation mechanism are built as an external level and are work on the input space of the data, in the following i will consider the architecture and the scope as the main source to present and categorize the mechanisms i have studied.

3.2 Architectures

We will now go on to look at a number of mechanisms for generating synthetic data, also going on to discuss them using the aforementioned applications and the previously defined taxonomy. In this section i won't include the Network architecture, which will have its own section because of its importance, and the domain-specific, because it can't be used in a general setting and is built ad hoc for each application.

3.2.1 Randomized mechanisms

The generation of synthetic data based on perturbation can be seen as:

$$x^s = x_i + \epsilon \tag{3.1}$$

Where ϵ is called **noise vector** obtained from a certain distribution. Some typical distributions to obtain ϵ are Gaussian and Laplace. Typically they are designed to have control over the maximum and minimum perturbation value to be obtained.

To maintain the distribution of data, we often use the technique of **masking**: let m be a vector where each element is 1 or 0 (usually generated with a Bernoulli distribution):

$$m = [m_1, ..., m_d] \in \{0, 1\}^d \tag{3.2}$$

We can then define a new sample using the product for each element of the vector, $\odot$:

$$x^s = (1 - m) \odot x_i + m \odot (x_i + \epsilon) \tag{3.3}$$

The idea is that for each feature of x_i, the corresponding value in m indicates whether the value will be preserved (value 0) or should be replaced (value 1). The replacement is done by adding the corresponding value in ϵ. Without modifying the whole sample, the distribution and properties of the data are better preserved.

We can notice how this mechanism all use a local scope: for generating a synthetic sample, only a subset of the training data is taken into consideration.

3.2.2 Mechanisms with probability functions

By estimating a multi-variable Gaussian distribution using the training set, we can sample on it to get new samples from the same class. This mechanisms are based on a global scope, trying to estimate the distribution of the training data.

One problem can be immediately noted: assuming that the data follow a Gaussian distribution greatly limits the use of this mechanism. Gaussian Mixture Model (GMM)-based mechanisms were able to overcome this problem, where more than one Gaussian is computed on the same class (where each Gaussian should identify a cohesive subset of the class).

Another approach is given by Kernel Density Estimation (KDE), where a kernel function is employed to estimate the distribution of data over each region. It's defined as the following:

$$\hat{p}(x) = \frac{1}{N+h}\sum_{i=1}^{N} K\left(\frac{x-x_i}{h}\right) \tag{3.4}$$

Where N is the number of samples in the dataset, h a smoothing parameter and K the kernel function. If the Gaussian kernel is employed then it is called Gaussian KDE.

From the same paper we have an example of different results by employing the above techniques for a dataset, as seen in Figure 3.1.

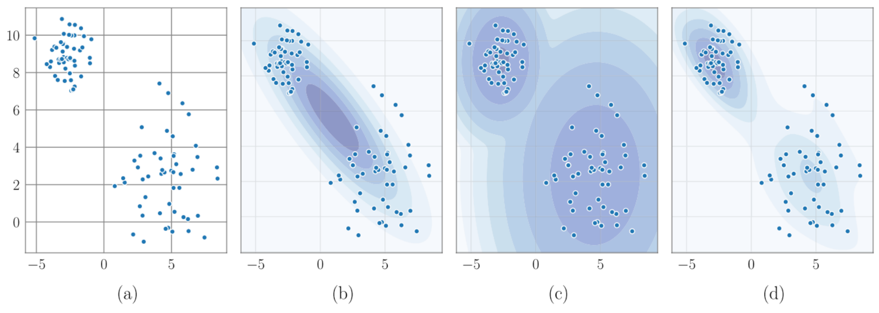


Figure 3.1: Examples of PDF mechanisms fitted to a dataset. (a) Original dataset, (b) Gaussian model, (c) Gaussian Mixture Model and (d) Gaussian Kernel Density Estimation

Taking advantage of Bayesian networks, we can construct Probabilistic Graphical Models (PGMs) as a collection of conditional distributions. Let us represent the distribution of D as a product of joint probabilities. If features have a conditional dependence relation, $(A|B)$, then features B are **parent feature** of A, $pa(A)$. We can then calculate the distribution as:

$$\mathbb{P}(x) = \Sigma_{v\in V}\mathbb{P}(x_v|x_{pa(v)}) \tag{3.5}$$

They are called graphical models because if you represent the features as nodes and the relation $pa(v)$ as all parent features pointing to v then you get an acyclic graph. From these graphs the synthetic data can be generated.

Bayesian networks can be used for synthetic data generation when the relationship between variables is known (or can be learned) and when the data is high-dimensional, not only making the sampling process non-trivial but also diminishing the range of possible applications.

3.3 Network architecture

The field of machine learning has seen remarkable advancements in recent decades, driven in large part by the introduction and rapid development of neural network models. One of the key advantages of neural networks is their remarkable capacity for feature extraction and representation learning. Neural networks can automatically learn relevant features from raw data, allowing them to uncover intricate patterns and relationships that were previously difficult to capture. This ability to learn powerful representations has enabled neural networks to achieve state-of-the-art performance on a wide range of tasks, including the field of generative models.

A first example of application of Neural Networks for generating synthetic data is Variational AutoEncorders (VAEs): a type of neural network that learns a representation of data in the form of a probability distribution. Within these networks there are two actors called **encoder** and **decoder**, which learn to project the data onto a new space. They are derived from ordinary AutoEncoders (AE), where the encoder and decoder learn an embedding representation of data by using a lower dimensionality space called the latent space. For this objective they use the reconstruction loss:

$$||x - \hat{x}||^2 \tag{3.6}$$

Where x is the original sample and $\hat{x}$ the sample reconstructed by the decoder. So the encoder will be optimized to include as much meaningful information of x in the latent space as possible. So AE are used in task like information compression or clustering. But since no constrains are posed to the latent space, we can not use the decoder to generate new samples since most of the latent space will not be mapped to meaningful data. The idea of VAE is to think about the latent space as a probability distribution. Here the used loss is the Variational loss:

$$E_z[\log p_\theta(x^i|z)] - D_{KL}(q_\phi(z|x^i)||p_\theta(x^i)) \tag{3.7}$$

Where q_ϕ represents the encoder and p_θ the encoder. The first term is used to train the decoder to reconstruct the input x^i from the latent space represented by z, while the second term force the latent space distribution (produced by the encoder) to be as close as possible to a normal distribution. This is known as the **Variational lower bound**: maximizing the above equation will produce a decoder able to approximate the distribution. Hence by randomly sampling from a multivariate normal distribution we can use the decoder to produce new synthetic data. Nevertheless the use of VAE has also been diminishing due to their limitation, specially in capturing details of the distribution, hence producing samples of lower quality compared to GANs.

Generative Adversarial Networks (GANs) are another type of Neural networks created for this purpose, where internally they are structured using two models: a generator and a discriminator. During the learning phase, the generator learns the distribution of the data while the discriminator estimates the probability of a sample being synthetic or from the original dataset. The purpose of the generator is to produce synthetic data as similar as possible to the original data such that the discriminator cannot distinguish it from the original data.

GANs were introduced by Goodfellow in his famous paper [16]. The main idea is in the training procedure: by taking inspiration from game theory, the encoder and decoder are trained using an adversarial, two players minmax game:

$$\min_G \max_D E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.8}$$

The first term represent the capacity of the discriminator to recognize a data as being in the dataset, while the second terms its capacity to recognize synthetic data. The discriminator is trying to maximize this term while the generator is trying to minimize it. This create a training procedure based on a loop where each component improves, up to the point where the samples of the generator are so similar to the synthetic samples that the discriminator can not improve anymore. At this point the procedure stop, and the generator will be used to generate synthetic samples. Before moving on one side note: initially the authors tried to train the generator by minimizing $\log(1 - D(G(z)))$. This proved to be difficult, since at the beginning the synthetic samples are too different and the discriminator was able to reject the samples with high confidence, thus saturating the gradient. To overcome this problem, in the actual implementation the generator is trained to maximize $\log(D(G(z)))$. This objective function results is the same dynamic, but provide much stronger and stable gradients.

Another important improvement for GANs were given by [23]. In the original paper of Goodfellow there was no mechanism of conditioning the generation process of the generator, hence randomly sampling from the normal distribution would provide random samples. In this paper the author showed that by just conditioning both discriminator and generator with an extra information y, one could condition the generation of both actors. The final game thus becomes:

$$\min_G \max_D E_{x \sim p_{data}(x)}[\log D(x|y)] + E_{z \sim p_z(z)}[\log(1 - D(G(z|y)))] \tag{3.9}$$

In practice for the generator the prior input noise z and the information y are combined in a joint hidden representation. The same is done for the discriminator, combining the input x and the information y.

One last improvement was brought in [2]: after GANs became popular it became more clear that one of their weakness relies on the training procedure. If the discriminator improves too much the generator won't be able to recover, and we will be stuck in a state called **mode collapse**, where at the end the synthetic samples will have an unacceptable quality. The authors shown that for the original GAN objective the training phase resulted in the minimization of the Jensen-Shannon Divergence. They claimed that changing the GAN to minimize the Wasserstein distance would improve the training for the following reason:

- Wasserstein Distance is continuous and almost differentiable everywhere.
- JS Divergence locally saturates as the discriminator gets better, thus the gradients becomes zero and vanishes.
- Wasserstein Distance as objective function is more stable than using JS divergence, thus the mode collapse problem is mitigated.

In practice the discriminator is changed, and to enhance this changing in the paper is now called critic. This new architecture was called Wasserstein GANs (WGAN), referring to the mathematical foundation it was derived from. The objective function of the critic becomes:

$$\max_{w \in W} E_{x \sim p_{data}(x)}[f_w(x)] - E_{z \sim p_z(z)}[f_w(g(x))] \tag{3.10}$$

To implement a minimization of the Wasserstein Distance the critic must have the property of Lipschitz continuity. This is done by clamping the weights to a small range ($[-e^{-2}, e^{-2}]$) in the paper. Later papers proposed different implementations, but **Gradient Penalty** [17] have become the standard for implementing a WGAN. Nowadays most GAN architectures uses a WGAN architecture with gradient clipping to ensure a stable training procedure.

GANs for tabular data

Tabular data introduces new aspects compared to other type of data:

- Complex relationship between features.
- Discrete numerical data.
- Categorical data.

Originally GANs were thought to be used only for data with a real-value domain, like images. Their application for tabular data was not immediate and brought a series of very interesting proposals.

medGAN [8] is our starting point, due to their attention to discrete data. One problems that emerges with tabular data is that the gradient can not be computed for discrete data, only for continuous. This becomes a problems for updating the generator. Their approach was based on leveraging an AutoEncoder: during the training the generator will generate samples in the latent space, then the samples will be reconstructed by the decoder before being given to the discriminator. The decoder of the AE is fine tuned while training the generator, hence it can be seen as an additional module of the generator. The architecture of the generator is a classical GAN architecture, with the additional use of residual network. Since the idea of medGAN is to produce synthetic medical data that can be freely shared they also started investigating possible attacks to GANs.

In the same year **Table-GAN** [25] was published. The biggest change compared to medGAN is that no latent space is used. Instead samples are converted in a matrix squared form, and both generator and discriminator use a Convolutional Neural Network (CNN). Since classical convolution is based on apply filters to captures relationships and semantic meanings between pixel, the same relationships and semantic meanings can be captured between features of tabular data. They also introduced two additional component in the loss of the generator: **information loss** is the discrepancy between the mean and the standard deviation of synthetic and real records, while **classification loss** is derived by training

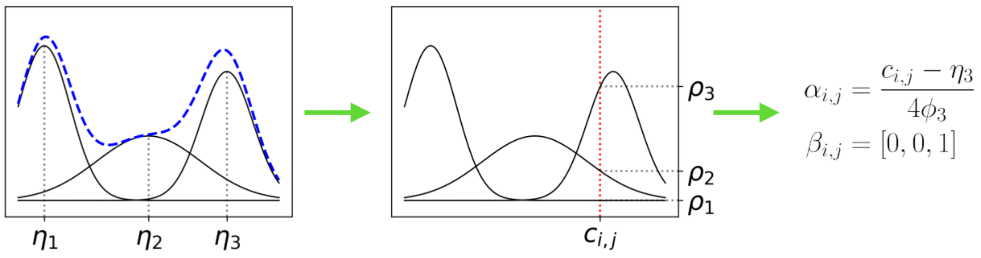

η_1
η_2
η_3
ρ_3
ρ_2
ρ_1
$c_{i,j}$
$\alpha_{i,j} = \frac{c_{i,j} - \eta_3}{4\phi_3}$
$\beta_{i,j} = [0, 0, 1]$

Figure 3.2: An example of mode-specific normalization [31]

a classifier on the training data and using it to classify the synthetic data. The introduction of these elements improved the quality of synthetic data: with the information loss the generator was enforced at better capturing the statistics of the data, while with the classification loss more semantic meaning were captured (for example a medical samples with the gender feature equal to female could not have anymore diseases that afflict only males).

The next jump in quality came with CTGAN [31], which was actually based on a model from the same author called TGAN [30] that used recurrent Neural Network. CTGAN employ a pre-processing phase for continuous features called **mode-specific normalization**: for each continuous features it estimates the number of modes using a Variational GMM [5] and fit a Gaussian. Then each value in the feature get normalized using the mean and standard deviation of the corresponding Gaussian, obtained by selecting the Gaussian with the higher probability. Thus each values is represented using a one-hot encoder representing the corresponding mode used, and the final scalar values. Figure 3.2 presents an examples, where the value η_3 is used to compute the processed value. Notice that $\beta = [0, 0, 1]$ because we use the third mode.
For Discrete data is also represented as one-hot encoding, but no pre processing is applied. They also proposed a solution to a known problem of GANs for tabular data: when a categorical column is unbalanced, in the training phase the values representing the minority will receive less attention from both generator and discriminator, hence the synthetic data will have a lower quality for specific data. Their solution is called **training-by-sampling**: using the conditional part of the GAN, during the training one value of one categorical features will be randomly selected, so for that for one specific loop the generator will be rewarded to generate samples respecting the condition, while the discriminator will receive the samples from the generator and rows from the training set that respect the condition. Note

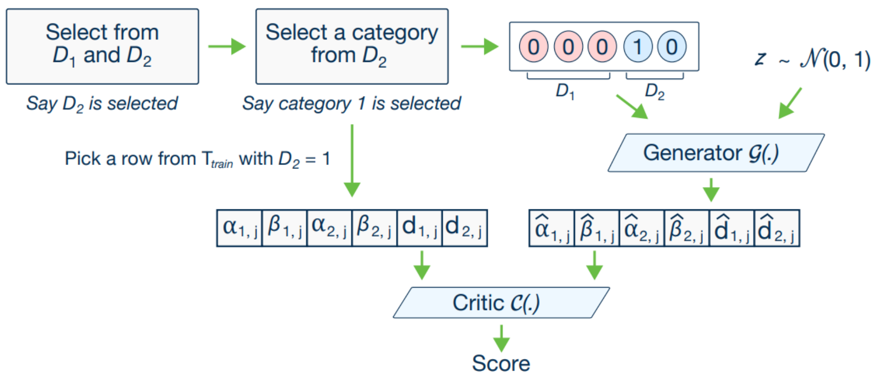

Select from D_1 and D_2
Say D_2 is selected
Select a category from D_2
Say category 1 is selected
0 0 0 1 0
D_1
D_2
$z \sim \mathcal{N}(0, 1)$
Pick a row from T_{train} with $D_2 = 1$
Generator $\mathcal{G}(.)$
$\alpha_{1,j}$ $\beta_{1,j}$ $\alpha_{2,j}$ $\beta_{2,j}$ $d_{1,j}$ $d_{2,j}$
$\hat{\alpha}_{1,j}$ $\hat{\beta}_{1,j}$ $\hat{\alpha}_{2,j}$ $\hat{\beta}_{2,j}$ $\hat{d}_{1,j}$ $\hat{d}_{2,j}$
Critic $\mathcal{C}(.)$
Score

Figure 3.3: An example of training-by-sampling [31]

that while the column have a uniform probability to be selected, the probability of selecting the values for the column will depends on their distribution (thus giving more chances to be selected to more frequent values). Figure 3.3 presents an example of training-by-sampling, where the first values of column D_2 is selected. One last improvement is given by using a discriminator with a WGAN architecture. Specifically they used the implementation based on gradient penalty [17].

The most recent and promising GAN for tabular data is **CTAB-GAN+** [35]. It was presented in an earlier version in [34] called CTAB-GAN. The version are similar in terms of adopting almost all of the above techniques: classification loss, information loss, training-by-sampling and mode-specific normalization. They were able to further improve the performances by defining **mixed variables**: continuous variables where there is a set of values with higher occurrences and a specific semantic meaning. For examples in a features representing the loan, the values 0 will have a specific semantic meaning, as 'this person does not have any current loans'. With the mode specific normalization this meaning may be lost. Thus they specified a modified version called **Mixed-type Encoder**, where for each continuous feature the user can specify a set of categorical value. Those values will not be affected from the pre-processing, thus the mode normalization will be applied only to the non-categorical values of the features. In the one-hot encoding for the features, the size will be equal to the number of identified modes plus the number of categorical values. Looking at Figure 3.4 we can see an example of mixed-type encoding, where the categorical values of μ_0 and μ_3 won't be processed, while the mode normalization is applied to μ_1 and μ_2. In total the one-hot encoding will have 4 positions: 2 for the modes and two for the

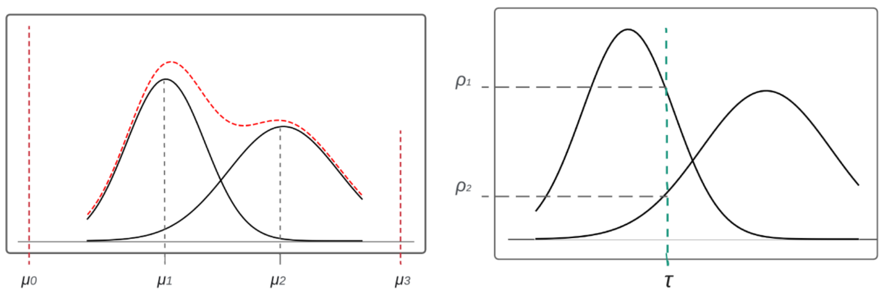


Figure 3.4: Mixed type variable distribution with VGM [35]

categorical values. We can notice that the computation of the two modes did not take into consideration μ_0 and μ_3.

With this new improvements CTAB-GAN was able to reach state-of-the-art results in many different datasets with complex relationship between features and mixed type distribution.

CTAB-GAN+ brought some improvements by adding the gradient clipping, thus using an architecture equivalent to a WGAN (as CTGAN), and adding an optional pre-processing called General Transform (GT), where the domain of the feature is encoded in the range of (-1, 1) using the maximum and minimum value to help the VGMM at recognizing the modes. The actual innovation of CTAB-GAN+ was the introduction of differential privacy, that will be discussed later.

Chapter 4

Attack and Defenses of Generation Mechanism

4.1 Attacking generation mechanism

Due to the increasing of popularity and use of synthetic data generation, many attacks have been proposed [32]. Some relies on some specific architectures (GAN or VAE) while others make no assumption on the underlying architecture. I have decided to work with an attack that makes no assumption on the underlying architectures, so i could compare the results using different generation mechanism. For this reason i chose the Membership Inference Attack (MIA): an attack where by only looking at the synthetic data, an attacker tries to infer some information about the training data. In this scenario the training data could not be shared due to the presence of sensitive attribute. The objective of the attack is to build a model able to answer the question: *is the following samples x present in the original training set?*

In this part i will explain the Membership Inference Attack, by using the technique presented in [7]. What we want to do in practice is define a function that given a sample will output 1 if it infers that the sample was in the training set and 0 if the sample was not in the training set. More formally:

$$\mathcal{A} = (x) \rightarrow \{0, 1\} \tag{4.1}$$

With a Bayes perspective, the attacker compute the probability of the sample belonging to the training set and if the probability is higher compared to the probability of not belonging to the training set we will return 1:

$$\mathcal{A} = \mathbb{1}\left[\log\left(\frac{P(x \in D_{train})}{P(x \notin D_{train})}\right) \geq 0\right] \tag{4.2}$$

How do we compute this probability? We can assume that a good generator will have approximated the training set distribution, so (by using $\mathcal{G}$ as the victim generator):

$$P_{D_{train}} \approx P_{\mathcal{G}} \tag{4.3}$$

$$P_{D_{train}}(x) \propto P_{\mathcal{G}}(x) \tag{4.4}$$

In a practical case this is still problematic: estimating the distribution can require a large number of samples that we don't have and since we don't have access to the latent space (the vector supplied to the generator) we can't control which samples are generated. Hence the author introduced Parzen window density estimation, were we can estimate the probability as:

$$P_{\mathcal{G}}(x) \approx \frac{1}{k}\sum_{i=1}^{k} e^{-L(x,\hat{x}^i)} \tag{4.5}$$

Where we can k generated samples denoted as $\hat{x}^i$ and L is a distance function. As the number of close samples increases the probability will tend to grow.

Lastly, to further simply our job we won't consider all the generated samples but only the closest one to our sample. Thus our $\mathcal{A}$ function will output 1 when we can find a sample which is close enough to x:

$$\mathcal{R}(x|G) = \min_{\hat{x} \in G} L(x, \hat{x}) \tag{4.6}$$

$$\mathcal{A}(x|G) = \mathbb{1}\left[L(x, \mathcal{R}(x|G)) \leq \epsilon\right] \tag{4.7}$$

Where G is the set of generated samples and ϵ the threshold. The Figure 4.1 is a good visual representation of what is happening (taken from the paper cited above).

Before moving on two things must be noticed: choosing ϵ requires a balance, being too restrictive would mean wrongly classify samples in the training set as not in the training set, while the opposite would arise if we have a bigger ϵ. Also

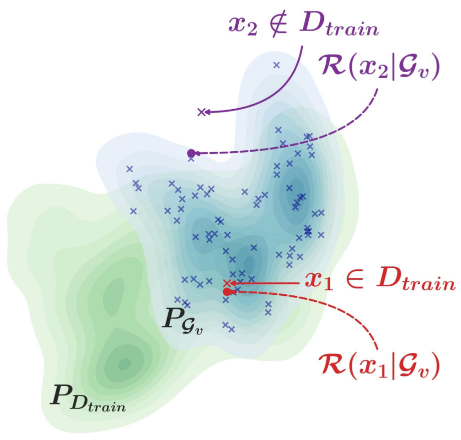


Figure 4.1: Attacks to GAN

the assumption of this attack is the key of its strength: as the generator becomes better at estimating the distribution this attack becomes easier. In fact we will see that there is a trade off between ML utility and Privacy.

4.2 Differential Privacy

To understand how different generation mechanisms (including CTAB-GAN+) try to prevents the MIA attacks we need to define what is differential privacy and how it's usually implemented in a GAN. This explanation is based on [27].

Let's define with M a classifier trained over a set of patients with cancer, called D. Since having a cancer (or have had it) is a personal information is very important that this information remains private. Let's now imagine we have a record that says "Bob has cancer" (we don't need to consider how it's actually built), and using M trained on D we get:

$$M_D(BOB) = 0.55 \tag{4.8}$$

Now we will add this record to the training set, written as M_{D+BOB} and let's consider two cases:

$$M_{D+BOB}(BOB) = 0.57 \tag{4.9}$$
$$M'_{D+BOB}(BOB) = 0.80 \tag{4.10}$$

We can imagine that in the first case the model improved because it now has access to more data, but in the second case the model is obviously overfitting and, worse, is leaking the Bob information that he has cancer.

Thanks to information theory we can measure the **privacy loss** as the logarithm of the ratio of the two values: $\log \frac{M_{D+BOB}(BOB)}{M_D(BOB)}$. Let's see it for the two cases:

$$\log \frac{0.57}{0.55} = 0.0357 \tag{4.11}$$
$$\log \frac{0.80}{0.55} = 0.375 \tag{4.12}$$

We have measured that the second case has 10 times more privacy loss compared to the first case!

We can now formally define **differential privacy** (DP): for any dataset D and for any set of outcomes S we try to minimize the privacy loss:

$$\log \frac{P(M_D \in S)}{P(M_{D'} \in S)} \tag{4.13}$$

Where D' is defined by removing one element from D. This computation is done by considering all possible samples, hence any configuration of D'.

We need to consider the situation where the privacy loss is equal to 0: in this case the model is not changing the output, thus the model is not learning from the added record. For this reason we will add some flexibility: now our objective is that the privacy loss is lower than an upper-bound ϵ called the **privacy budget**:

$$\log \frac{P(M_D \in S)}{P(M_{D'} \in S)} \leq \epsilon \tag{4.14}$$

This will be our trade-off between how much we learn from each sample and the privacy of each sample. This version of DP is called ϵ-differential privacy and is usually written as:

$$P(M_D \in S) \leq e^{\epsilon} \cdot P(M_{D'} \in S) \tag{4.15}$$

If we are able to find such a value ϵ for a machine learning model M we have a formal privacy guarantee for any training sample in M.

This is still very tight: we will introduce a parameter δ which represent a probability that the above equation is not satisfied for some cases. We will then extend the formula and get the (ϵ,δ)-differential privacy:

$$P(M_D \in S) \leq e^{\epsilon} \cdot P(M_{D'} \in S) + \delta \tag{4.16}$$

By reducing the value for both ϵ and δ with a good probability we can guarantee for each sample in D an acceptable level of privacy (enough to make attacks very difficult) while keeping ML utility high enough.

4.3 Implementing Differential Privacy

Now that we have a formal definition of differential privacy we need a way to achieve it. The main idea is to add noise: the more noise we add the more the model will learn differently the training data and the more privacy we are guaranteeing.

[1] proposed to apply differential privacy during the training phase, into the stochastic gradient descendent, known as **DP-SGD**. This has an interesting advantage over adding noise to the output: while in the former case the privacy guarantee deteriorates as we produce outputs (this is known as the basic composition theorem [19]: if you interact T times with a mechanism with (ϵ,δ)-DP you will end up with a ($T\epsilon$,$T\delta$)-DP), by using DP-SGD we can use the model as long as we want without worrying about privacy.
DP-SGD works like a normal SGD but we add two steps after computing the gradient and before applying it:

1. **Clip the gradient**: divide the gradient by it's norm divided by the parameter C, thus ensuring that its norm is at most C. In this way we introduce a bound to how much the model is learning from the batch.

2. **Add noise**: add noise to the gradient by using a Normal distribution with standard deviation $C\sigma$, where σ is the **noise multiplier**.

In their paper they proposed the following configuration: the hyper parameter C can be set equal to 1, while σ will depend on the (ϵ,δ)-DP by the following formula:

$$\sigma = \sqrt{2 \cdot \log \frac{1.25}{\delta}} / \epsilon \tag{4.17}$$

With this configuration it's guaranteed (ϵ,δ)-DP at each individual step in the algorithm.

But can we measure the DP for the **overall** SGD algorithm? We need to consider the following statement:

- With appropriate value we can guarantee (ϵ,δ)-DP at each step.
- Due to the basic composition theorem doing this T times we are left with ($T\epsilon$,$T\delta$)-DP.
- privacy amplification theorem [20]: if a model M is (ϵ,δ)-DP for a training set D, then if we train it using a data set D' obtained by sampling from D a fraction q of data then we have ($q\epsilon$,$q\delta$)-DP.

Adding these together we can thus see that our DP will have an upper-bound of ($qT\epsilon$,$qT\delta$)-DP. This has a critical point: the privacy budget and probability of failure increases as the number of iterations T increases. But in the same paper where the algorithm was proposed an analysis to get a tighter bound was done.

They used **Moments accountant**, a technique that permitted to prove that the actual overall differential privacy becomes $(\mathcal{O}(q\sqrt{T}\epsilon), \delta)$-DP, thus the failure probability is always the same and the privacy budget increase is now proportional to $\sqrt{T}$. We can use the batch size to adapt q to prevent ϵ to increase too much. CTAB-GAN+ uses the above implementation of DP-SGD during the training when we specify the ϵ and δ parameters, otherwise it will proceed using a normal SGD procedure.

Chapter 5

Experiments

This chapter will use different mechanisms for generating synthetic data, with the objective of exploring their capabilities and deeper understanding the true gains of using generative methods based on a network architecture. For each section i will propose a research question and presents some experiments to try to answer and present novel results.

My analysis include CTAB-GAN+ [35] and other algorithms, categorized with the main architecture of synthetic data generations:

- Gaussian Mixture Models (GMM) for probabilistic architectures.
- SMOTE and Mixture will be used for representing algorithms based on perturbation.
- For network based technique i will employ CTGAN.

I initially started trying to also employ Table-GAN, but the results for the oversampling part did not satisfy my expectations: at some points of the training phase there would be a convergence failure and the synthetic data started looking like random noise. Since i wanted to compare CTAB-GAN+ with another GAN-based approach but i couldn't used Table-GAN i decided to use CTGAN [31], which is similar to CTAB-GAN+ and can be seen as less complex approach (since it doesn't include DP-SGD or classification loss).

Comparing Table-GAN and CTGAN there are two points to consider:

- CTGAN implements a gradient penalty during the training, implementing a WGAN. It has been shown that this helps the training process.

- CTGAN uses a pre-processing phase, where the data is normalized according to a GMM fitted for each column. Also this approach has been shown to help the training process.

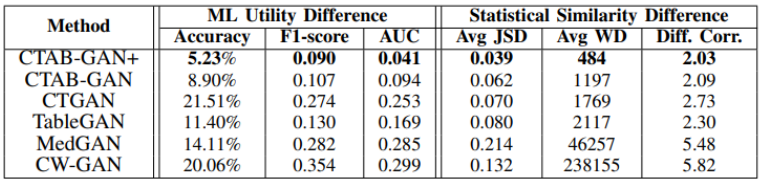
My thesis is based on 3 main evaluations for synthetic data. For each one i will introduce a dataset, explain the purpose and how to measure the performance for each generation algorithm:

- Oversampling: starting with a very unbalanced dataset, i want to add samples of the minority class to balance the dataset.
- Privacy: a very important application, where i want to create a new dataset of synthetic data and share it without any problem of privacy.
- Fairness: can i use the synthetic data to augment the training set and avoid unfairness in the decision boundaries?

An important introduction of CTAB-GAN+ is **differentially-private stochastic gradient descent** (DP-SGD) where we clip the gradient to our clipping norm C and add noise base on a noise multiplier σ. These parameters will be used in the Privacy section while for the Oversampling and Fairness sections i won't limit the gradient vector (since in these cases we are not in a situation where privacy must be guaranteed).

5.1 Oversampling

The improvement of synthetic data for tabular data has been huge thanks to the introduction of new techniques based on Generative Adversarial Networks. But there still is some problems in terms of measuring the improvement: most datasets are based on using categorical data, the results that the authors of CTAB-GAN+ presented data with mostly categorical data as seen in Figure 5.1. With this setting is impossible to compare GANs with more traditional and reliable techniques like SMOTE and GMM. Thus i will introduce a dataset with only continuous columns, with the objective of being able to compare traditional oversampling techniques with two network based, CTGAN and CTAB-GAN+. My research question is: *will GANs-based approaches overcome traditional models*?



Method	ML Utility Difference			Statistical Similarity Difference		
	Accuracy	F1-score	AUC	Avg JSD	Avg WD	Diff. Corr.
CTAB-GAN+	**5.23%**	**0.090**	**0.041**	**0.039**	**484**	**2.03**
CTAB-GAN	8.90%	0.107	0.094	0.062	1197	2.09
CTGAN	21.51%	0.274	0.253	0.070	1769	2.73
TableGAN	11.40%	0.130	0.169	0.080	2117	2.30
MedGAN	14.11%	0.282	0.285	0.214	46257	5.48
CW-GAN	20.06%	0.354	0.299	0.132	238155	5.82

Figure 5.1: Results of GANs on tabular data [35]

5.1.1 Dataset analysis

We can start by analysing our chosen dataset: HTRU2 [22]. This dataset include a collection of measurement for Pulsars (a rare type of Neutron star). The objective is to be able to distinguish samples of signals that are generated from a Pulsar compared to those generated by noise. This dataset is interesting because only roughly 9% of records are generated by Pulsars, the vast majority is from noise.

Let's look at some samples in Table 5.1. Since the same 4 four statistics (mean, standard deviation, excess kurtosis and skewness) are computed for the two sources (Integrated profiles and DM-SNR curve) i decided to use those as columns and the two sources as sub-columns.

Mean		Std Deviation		Excess Kurtosis		Skewness		
Int Prof	DM SNR	Int Prof	DM SNR	Int Prof	DM SNR	Int Prof	DM SNR	Label
140.56	55.68	-0.23	-0.70	3.20	19.11	7.98	74.24	0
99.37	41.57	1.55	4.15	27.56	61.72	2.21	3.66	1
27.77	28.67	5.77	37.42	73.11	62.07	1.27	1.08	1

Table 5.1: Statistical Measures for Different Categories

Since our objective is to create synthetic data belonging to the Pulsar group this analysis will take into account samples where the labels is equal to 1. Let's start by looking at the correlation map: Figure 5.2. As we can expect, there is an higher correlation between variables computing statistics from the same source (integrated profile and DM-SNR curve), while for different sources the correlation diminishes.

For the single variables, all of them resemble a Normal distribution or a negatively skewed distribution as we can see in Figure 5.4. By plotting some scatter plots we can observe some interesting shapes, like in Figure 5.3, where we can assume there is a potential non-linear relationship between the two variables. Such

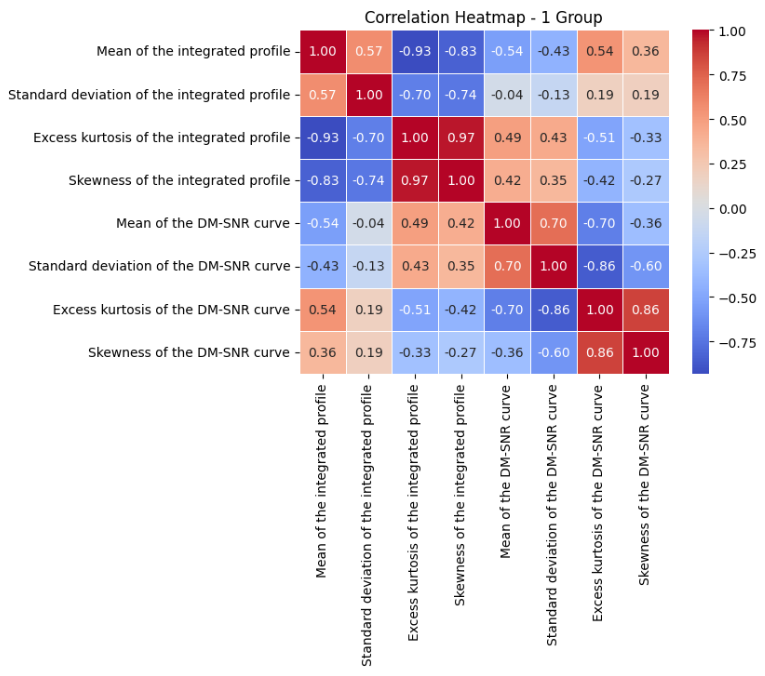
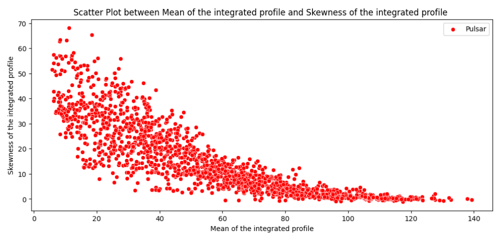
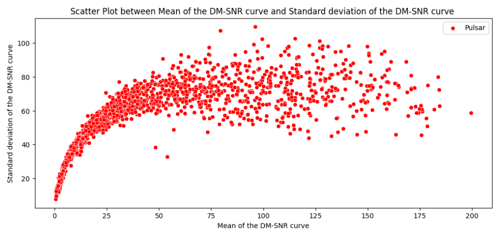


Figure 5.2: Correlation map

a pattern is often observed in natural or social phenomena, where the relationship between two variables is not strictly linear, but rather exhibits a more nuanced, non-linear behavior.



(a) Mean and Skewness of Integrated Profile

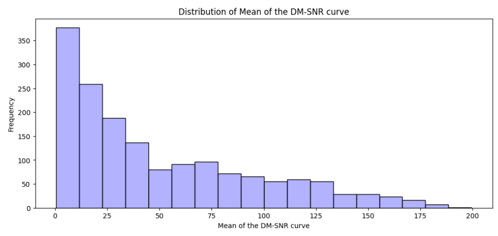


(b) Mean and Std of DM-SNR curve

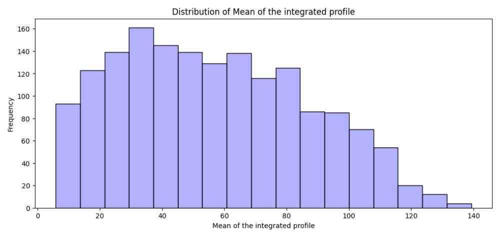
Figure 5.3: Plots for Pulsars

5.1.2 Algorithms Application

To test our algorithm i split the data, where 20% will be used as a test set and the remaining as training set (to produce the synthetic data and to train the classifier). Since we started with only 1639 possible cases for Pulsars the samples



(a) Statistic Plot 1



(b) Statistic Plot 2

Figure 5.4: Histograms for Pulsars

of positive samples in the test set is too low for employing accuracy. A commonly used metric is F_β score, which takes into account precision and recall:

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{5.1}$$

I decided to measure the performances by using $\beta = 2$, which gives an higher weigh for the recall value. In this way i can measure if the algorithm is correctly classifying as many Pulsars as possible, even if those are the minority in both train and test set. For the classifier, i employed a Random Forest Classifier, specifically the implementation of sklearn. For each algorithm i added synthetic data until the ratio of classes for the training set was balanced. In Table 5.2 we can see the best results. For Gaussian Mixture Model i tried two approached: the first one i trained GMM in the raw data, while in the second approach i applied PCA (so the generated samples needed to be projected to the original shape). We can see that the results are similar in terms of F_2 score, but i will keep applying both approches later.

For CTAB-GAN+ i trained it in two configurations: in the first one i trained it using both samples from Pulsars and Noise, then i filtered the synthetic data selecting only Pulsars synthetic samples, while in the second I trained the model using using only Pulsars samples. Currently in the state-of-the-art there is no clear advantage for the oversampling scenario: the former may be helpful at understanding the boundaries of the classes, but with the latter the model focus on the useful class. As an example [12] obtained the best results by training a GAN only on minority samples.
Regarding the noise class during the analysis i also computed the F_2 scores for that class: with the synthetic data the value will decrease but it never went lower than 97%. The reason why the value is decreasing relies on the algorithm

	Without	GMM PCA	GMM	SMOTE	CTAB GAN+$_{all}$	CTAB GAN+$_{Pulsars}$
Depth 5	0.876	0.896	0.887	0.895	**0.900**	0.894
Depth 6	0.885	0.898	0.891	0.893	0.896	0.896
Depth 7	**0.891**	0.900	0.896	0.897	0.897	0.896
Depth 8	0.882	0.897	**0.903**	**0.898**	0.896	0.899
Depth 9	0.885	0.897	0.899	0.895	0.894	**0.904**
Depth 10	0.889	**0.901**	0.900	0.895	0.895	0.897

Table 5.2: Results for oversampling

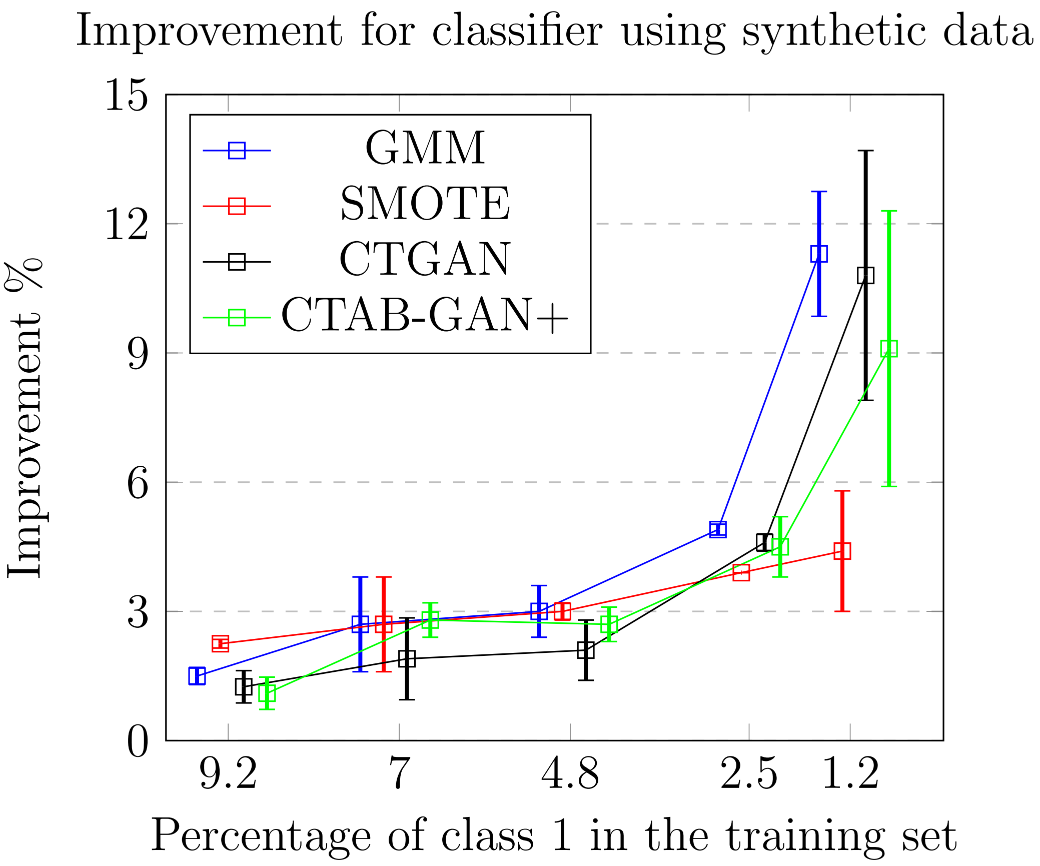
focusing on both class and not classifying the majority of samples as Noise, hence the recall of the Noise class is a bit lower now.

	Without	CTAB GAN+$_{all}$	CTAB GAN+$_{Pulsars}$	CTGAN$_{all}$	CTGAN$_{Pulsars}$
Depth 5	0.876	**0.900**	0.894	0.882	0.893
Depth 6	0.885	0.896	0.896	0.888	0.893
Depth 7	**0.891**	0.897	0.896	0.892	0.892
Depth 8	0.882	0.896	0.899	0.897	0.894
Depth 9	0.885	0.894	**0.904**	**0.900**	**0.894**
Depth 10	0.889	0.895	0.897	0.898	0.890

Table 5.3: Results for GANs

Comparing the two GAN-based models in Table 5.3, not only CTAB-GAN+ outperforms CTGAN in both configurations, but interestingly CTGAN prefers to have both labels during training, compared to CTAB-GAN+ that got the best results with the training set composed of only Pulsars. This is probably due to the possibility of using pre-processing methodologies, specific to the Pulsars class. Some columns have a mixed type variable only for the Pulsar class, thus can be used only the setting with only Pulsars.

In some context (for example fraudulent transaction identification) a ratio of 9% of one class is not considered too unbalanced. We can also see that the improvement given by using CTAB-GAN+ is only equal to 1.3%. I decided to repeat the above experiment multiple time, keeping track of the improvement gained by using synthetic data from each generator, while also randomly removing samples from the minority class. Results as the ratio of Pulsars samples decrease is shown in Section 5.1.2, where i put together the results for GMM with and without PCA and the different configurations of the GANs. The increase in accuracy is shown in percentage for visualization purpose. For summarizing the 6 experiments for each configuration, i used the median and the first and third quartile. In this way i could present the results without any shift due to a better or worst value while also showing the spread. As the percentage of Pulsars samples diminishes, the performances of synthetic data varies.



As the number of samples diminishes, for GMM the best results were obtained by applying PCA before, contrarily to what happens in the situation where we don't remove samples. For both GANs the best results were obtained when training only with data coming from the Pulsars. Notable CTGAN is the generator that has more difficulties at learning the data distribution but has a peak as the number of positive samples diminishes, while CTAB-GAN+ can keep up the pace. This is explained by the pre-processing of the data, that amplifies the data distribution and avoids that CTAB-GAN+ overfit. A known behaviour that is confirmed in this experiment is that as the number of samples diminishes the improvement of SMOTE starts is less strong compared to others.

By considering other factors such has computation time and the requirements of specific hardware, GMM still remains the best methodology for this setting.

CTGAN was able to outperform it in some scenarios but only for a small percentage and requiring a longer time to train.

5.2 Privacy

Privacy in the context of synthetic data refers to the possibility of freely sharing a dataset containing sensitive attributes, since with synthetic data the samples are not directly derived from real people. As discussed above, attacks to synthetic data like the Membership Inference Attack (MIA) may take advantage of this assumption, deriving membership of samples in the training data by only looking at the synthetic data. In this section i will investigate how each model is able to withstand the attack by performing it and measuring the accuracy, thus my research question is: *are GANs approaches better at withstanding the MIA attack*?. Since an accuracy close to 0.5 means that the model is basically classifying all samples in one class, we can assume that it is not able to understand which were the samples in the training set used for generating the synthetic data.

5.2.1 Dataset analysis

I decided to use the Adult dataset [4], since it contains a personal information about an individual: whether he earns more or less than 50k per year. The dataset is also known to be very unbalanced, favoring some individuals regarding who earns more than 50k (usually white male individuals) [15], but this is beyond the analysis for this section.

Before performing a MIA attack and evaluate each algorithm based on the resistance to the attack, let's see the attributes and some example of samples. Here we can see each attribute with a short description:

1. **Age:** Integer representing the age of the individual.
2. **Workclass:** Categorical feature describing the type of employment.
3. **Fnlwgt:** Integer feature representing the final weight, which is the number of units in the target population that the responding unit represents.
4. **Education:** Categorical feature representing the education level.

5. **Education-num:** Integer representing the numerical encoding of the education level.
6. **Marital-status:** Categorical feature describing the marital status.
7. **Occupation:** Categorical feature describing the occupation.
8. **Relationship:** Categorical feature describing the relationship status.
9. **Race:** Categorical feature describing the race.
10. **Sex:** Binary feature representing the gender.
11. **Capital-gain:** Integer feature representing the capital gains.
12. **Capital-loss:** Integer feature representing the capital losses.
13. **Hours-per-week:** Integer representing the number of hours worked per week.
14. **Native-country:** Categorical feature describing the native country.
15. **Income:** Target binary feature representing income levels, with categories '>50K' and '$\leq$50K'.

final_weight (fnlwgt) tells the proportion of the population that has the same set of features. Basically, every row in the original table was de-duplicated and final_weight stores the number of rows that have exactly the same value. Also Education-num is the integer representation of the Educational level, hence Education-num and Education have a one-to-one correspondence. For this reasons the columns fnlwgt and Educational have been removed before starting the experiment.

Here we can see a sample belonging to the training set:

Age: 56
Workclass: Local-gov
Education-num: 13
Marital-status: Married-civ-spouse
Occupation: Tech-support
Relationship: Husband
Race: White

Sex: Male
Capital-gain: 0
Capital-loss: 0
Hours-per-week: 40
Native-country: United-States
Income: >50K

After analyzing the dataset i decided to apply the following modifications:

- Some columns use different values for unknown values: work class, native country and occupation uses the question mark, while capital gain uses 99999. Those values are substituted with NaN.
- Since the creation of synthetic data requires a dataset with a good quality, rows with at least one NaN will be removed from the dataset.
- Around 11% of the dataset is composed of duplicates, that will be removed before continuing to avoid bias.

To simulate the attack, I sampled two subsets of the original data: one subset is removed from the original data and consists of samples where I expect the attacker to return zero, while the other subset consists of samples that are also in the training data and thus I expect the attacker to recognise them. Both sets are balanced with respect to the target feature. The MIA attack requires a distance function L, to measure the distance between samples. It's computed with the euclidean distance, where for computing the distance for each feature i used the following strategies:

- For continuous and numerical features i normalized the values, so that the range of each variable is between 0 and 1. Hence for each variable the distance can be computed by using the euclidean distance.
- For categorical features since a notion of distance is usually wrong i simply returned 1 when the values are different and 0 if they are equal.

As a comparison method i used a **Mixup** model, where a synthetic samples x^s is obtained by mixing up two samples from the data:

$$\begin{aligned} x^s &= \lambda x_i + (1-\lambda)x_j \\ y^s &= \lambda y_i + (1-\lambda)y_j \\ \lambda &= Beta(\alpha, \alpha) \end{aligned} \tag{5.2}$$

Two version of Mixup have been used: in one case the λ values are close to 0, trying to keep statistical similarity. In the second case lambdas will tend to 0.5, so most samples will contains an average of the samples loosing the statistical similarity while trying to minize the accuracy of the attack.

Since Differential Privacy is based on adding noise to the data i will also keep track of statistical similarity between the original dataset and the synthetic ones. The idea is to understand which generation mechanism can resist the attack while keeping he statistical information. I have used the following metrics (the same used by [35]):

- **Jensen-Shannon divergence** (JSD): used to quantify the difference between the probability mass distributions of individual categorical and numerical variables.
- **Wasserstein distance** (WD): used to capture how well the distributions of individual continuous/mixed variables are emulated by synthetically produced datasets in correspondence to real datasets. In our case the only variable with a continuous behavior is capital_gain.
- **Difference in pair-wise correlation** (Diff. Corr.). To evaluate how well feature interactions are preserved in the synthetic datasets. For the correlation Pearson correlation is used for continuous, while Theil uncertainty coefficient is used to measure the correlation between any two categorical features.

Let's start by looking of the accuracy of the attack for each mechanisms in Table 5.4, since i performed multiple times the attack (due to the sampling process being random) each row represents the outcome for a single experiment. Then for the two GANs i computed the above statistical metrics to compare the attack accuracy and the similarities of the data, in Table 5.5 we can see the results. While doing this computation i noticed that CTGAN produced some samples where the values for capital_gain and capital_loss were not respecting the constraints: those values cannot be negative, but since there is no classification loss (compared to CTAB-GAN+) the GAN is not learning it. I decided that for each synthetic dataset before computing the accuracy and statistics i pre-process the data to correct unacceptable data (negative values for capital, underage samples). Lastly: for the fifth case CTAB-GAN+ was generating an high number of synthetic samples identical to original samples. To avoid this situation of overfitting i applied differential privacy with an high privacy budget ($\epsilon = 1$), avoiding the overfitting problem without interfering too much with the training.

	Mixup	Mixup$_{privacy}$	CTGAN	CTAB-GAN+
1	0.6875	0.601	0.5125	0.507
2	0.7025	0.617	0.514	0.510
3	0.6855	0.605	0.505	0.505
4	0.696	0.607	0.5215	0.525
5	0.691	0.6055	0.51	0.5035
6	0.692	0.6195	0.5215	0.511
mean	0.6924	0.6092	0.5141	**0.5103**
median	0.6915	0.6063	0.5133	**0.5085**

Table 5.4: Attacks accuracy

	CTGAN				CTAB-GAN+			
	JSD	WD	Diff Corr	copies	JSD	WD	Diff Corr	copies
1	0.1112	0.0206	0.5983	1216	0.0861	0.0211	1.4435	1253
2	0.1122	0.0210	0.5808	816	0.0827	0.0227	1.3330	1536
3	0.0997	0.0272	0.6188	1348	0.0897	0.0271	1.3668	1212
4	0.1111	0.0157	0.634	788	0.0949	0.0320	1.3821	964
5	0.1179	0.0093	0.6426	848	0.0864	0.0269	1.3473	1293
6	0.1145	0.0362	0.6012	892	0.0963	0.0373	1.3890	1055
mean	0.1111	**0.0217**	**0.6126**	**985**	**0.0894**	0.0279	1.3770	1128
median	0.1117	**0.0208**	**0.6100**	**870**	**0.0881**	0.0270	1.3745	1233

Table 5.5: Statistic similarity between synthetic and original data

The data generated by the two GANs are able to avoid the attack, by bringing it close to 0.5 (where the attacker is essentially randomly guessing). Also CTAB-GAN+ is a little more resistant compared to CTGAN. The results of Table 5.5 are not really explaining why the attack has this difference, since it seems like CTAB-GAN+ should be weaker due to the number of copies. By analyzing the synthetic data i noticed that the number of duplicates in the copies is high (for some real data CTAB-GAN+ have created not one but two or more synthetic data) and in general both GANs have some duplicates (this is probably due to the fact that the statistical constraints of the data force the generator to create similar data). In Table 5.6 and Table 5.7 i compute the statistics after removing the duplicates: this will ensure that we can measure more accurately the statistical similarity and the number of duplicates. We can also notice that the results for the MIA attack won't change, since duplicates don't add any meaningful information for finding the closest synthetic sample. The final results show a better similarity for JSD for CTAB-GAN+ but at the cost of more duplicates compared to CTGAN.

	CTGAN				
	JSD	WD	Diff.Corr.	copies	duplicates
1	0.1119	0.0204	0.66	841	773
2	0.1135	0.0216	0.5807	657	609
3	0.1010	0.0282	0.6231	917	847
4	0.1117	0.0157	0.6334	639	423
5	0.1187	0.0095	0.6421	618	575
6	0.1153	0.0358	0.6040	732	534
mean	0.1120	**0.0204**	**0.6239**	**734**	**627**

Table 5.6: Statistic similarity between synthetic and original data without duplicates

	CTAB-GAN+				
	JSD	WD	Diff.Corr.	copies	duplicates
1	0.0886	0.0216	1.4452	874	1463
2	0.0855	0.0229	1.3328	1001	1566
3	0.0924	0.0281	1.3618	821	1427
4	0.0969	0.0328	1.3906	717	979
5	0.0888	0.0277	1.3550	955	1040
6	0.0997	0.0395	1.3844	599	1652
mean	**0.0920**	0.0255	1.3783	828	1355

Table 5.7: Statistic similarity between synthetic and original data without duplicates

Before moving on it's worth trying to explain why there is such a big gap between the two Mixup generators and the two GANs. Taking inspiration from

[35] i computed **Distance to Closest Record** (DCR) and **Nearest Neighbour Distance Ratio** (NNDR) between synthetic data and real data. Both gives an insight on how much the synthetic data is too close to the original data and how much information it can be gathered by attacking it. We can observe in Table 5.8 the correlation between accuracy and distance: especially when NNDR increase the MIA attack accuracy decrease. I also included the test set as a maximum threshold: since the test set is not derived from the training set, it can be interpreted as a set with good similarity that maximize the distance between samples. In this way we can derive the following result: CTAB-GAN+ has the best value for both metrics, since for DCR it has the highest value, while for NNDR it has the closest value to the test set, hence the synthetic samples are not too distant from the original samples.

	Mixup		Mixup$_{privacy}$		CTGAN		CTAB-GAN+		Test set	
	DCR	NNDR	DCR	NNDR	DCR	NNDR	DCR	NNDR	DCR	NNDR
1	0.2214	0.3244	0.3570	0.5214	0.3715	0.6066	0.3817	0.6060	0.9473	0.5744
2	0.2206	0.3087	0.3686	0.5155	0.3561	0.6009	0.4121	0.6075	1.0437	0.6052
3	0.2221	0.2988	0.3761	0.5123	0.4046	0.5877	0.4406	0.6251	1.0130	0.6029
4	0.2140	0.2903	0.3685	0.5158	0.4322	0.6328	0.4394	0.6229	1.0023	0.6074
5	0.2166	0.3149	0.3807	0.5063	0.4385	0.6391	0.3994	0.5891	0.9203	0.5814
6	0.2122	0.3027	0.3662	0.4986	0.4416	0.6409	0.4495	0.6230	1.0608	0.5923
mean	0.2178	0.3066	0.3695	0.5117	0.4074	0.618	**0.4205**	**0.6123**	1.0126	0.5939
median	0.2186	0.3057	0.3686	0.5139	0.4184	0.6197	**0.4258**	**0.6152**	1.0077	0.5976

Table 5.8: Measure of closeness between synthetic and original data

For completeness purposes i decided to train both GANs while increasing the number of epochs (from 150 to 250). In this way i expect both synthetic dataset to be closer to the original data while being more vulnerable to the MIA attack. For the CTAB-GAN+ samples, i kept using the privacy budget for avoiding overfitting.

From Table 5.9 and Table 5.10 seems like more epochs improved CTAB-GAN+ on all metrics, while for CTGAN it diminished the number of copies and duplicates but worsened the statistical similarity. In the Table 5.11 we can see the results of the MIA attack for the newly created synthetic datasets. As expected the accuracy of the attack for all cases has improved or remained the same.

Since it looks like CTAB-GAN+ is benefiting a lot from the increase of the epochs i tried retraining it for the same datasets but using a number of epoch

	CTGAN				
	JSD	WD	Diff.Corr.	copies	duplicates
1	0.1137	0.0238	0.6082	475	217
2	0.1134	0.0271	0.5785	415	172
3	0.1161	0.0267	0.6938	875	487
mean	0.1144	0.0259	**0.6283**	**588**	**292**

Table 5.9: Statistic similarity between synthetic and original data without duplicates and with 250 epochs

	CTAB-GAN+				
	JSD	WD	Diff.Corr.	copies	duplicates
1	0.0819	0.0217	1.0915	982	897
2	0.0833	0.0140	1.1991	1091	89
3	0.0862	0.0321	1.1791	859	1816
mean	**0.0838**	**0.0226**	1.1566	977	934

Table 5.10: Statistic similarity between synthetic and original data without duplicates and with 250 epochs

equal to 350 and keeping the privacy budget (to avoid overfitting). Interestingly we can observe from Table 5.12 that not only the statistical similarity improves but also the resistance to the attack. This is due to the generator learning even better the distribution of the data, thus being able to generate samples enough sparse.

5.2.2 Half-size dataset

Lastly we can test for all generators what happens when the size of the training data is reduced: as [7] showed a reduction in the size of the training data usually implies a weaker synthetic dataset to the MIA attack, due to the higher changer of privacy leakage. Mixup generators have a higher accuracy for the attack (around 73% for Mixup and 62% for Mixup$_{privacy}$). It's important to test higher epochs, due to the lower availability of data the generators need more iterations to better understand the data and avoid overfitting. We can observe from Table 5.13 that an higher number of iterations implies a lower attack value. Also GANs can still withstand the attack, thus proving more robust compared to other methods.

Given that CTAB-GAN+ appears to face a slightly higher accuracy with the

	CTGAN	CTAB-GAN+
1	0.5125	0.513
2	0.5145	0.515
3	0.5145	0.506
mean	0.5138	**0.5113**

Table 5.11: Attacks accuracy with 250 epochs

	CTAB-GAN+					
	JSD	WD	Diff.Corr.	copies	duplicates	attack acc.
1	0.0724	0.0159	1.1012	1284	1478	0.508
2	0.0791	0.0152	1.0907	1157	1111	0.507
3	0.0749	0.0254	1.0730	1078	1365	0.512

Table 5.12: Statistic similarity between synthetic and original data without duplicates and with 350 epochs

attack, i propose employing differential privacy as a solution. By constraining the privacy budget, we aim to mitigate the risk of overfitting and diminish the accuracy of the attack. As demonstrated in Table 5.14, the synthetic data generated by CTAB-GAN+ with the integration of differential privacy exhibits the highest resilience against the attack. This underscores the efficacy of incorporating it into a synthetic data generation pipeline.

5.3 Fairness

In this chapter i want to focus on the application of synthetic data for leveraging possible discrimination scenarios, where a machine learning model has learnt a set of parameters that causes discrimination against a category of people. Two families of technique have been used: Explainable AI (XAI) techniques for machine learning models and fairness metrics. The former tries to identify the sources of bias of a model by explaining its reasoning, while the latter uses a set of metrics to evaluate the unfairness of a model. The actual use of XAI for fairness has been criticized [10], since it has been shown that some biases are not always captured and we have the risk of *fairwashing*, where the users are mislead into trusting biased or incorrect models. Still i will apply this techniques to have a description on how the model is internally reasoning.

My research question regards the impact of synthetic data for the reduction

	CTGAN_{200}	CTGAN_{250}	CTAB-GAN+_{200}	CTAB-GAN+_{250}
1	0.522	0.517	**0.5155**	0.5195
2	0.51	**0.5075**	0.511	0.523
3	**0.5135**	0.514	0.5155	0.516

Table 5.13: Attacks accuracy with half size dataset

	best CTGAN	best CTAB-GAN+	$\text{CTAB-GAN+}_{privacy}$	privacy budget
1	0.517	0.5155	0.514	10^{-3}
2	0.5075	0.511	0.5075	10^{-4}
3	0.5135	0.5155	0.5125	10^{-5}

Table 5.14: Attacks accuracy with half size dataset, including differential privacy

of risk of bias, more specific: *Will introducing synthetic data in the training set diminish the risk of bias?*

5.3.1 Explainable AI

I will use the Adult dataset, pre-processed like in the Privacy section. For the sake of simplicity i also removed the feature marital_status, due to its similarity with the feature relationship. Then I divided the data into training sets and test sets, and i trained an interpretable model. The choice of the model being **interpretable** gives me the possibility of analyzing it using model agnostic and model specific explanation method, to check if i can find any possible discriminant scenario. For the Adult dataset i found a decision tree to be the best performing interpretable model.

Since the decision tree reached a depth equal to 8 it's impossible to visualize it, but we can employ an explanation method for decision tree known as **feature importance**: it measures the importance of each feature for the model with an **Impurity-based feature importance**, thus the importance of a feature is the (normalized) total reduction of the impurity criterion obtained by using that feature for splitting. It quantifies how much each feature influences the decisions made by the tree during training. Higher importance values indicate features that have a stronger impact on the model's predictions. Table 5.15 shows that the main features used by the model to classify are *relationship*, *capital_gain* and *education_num*.

Another methodology commonly used for measuring the impact of each feature in the decision process of any model is **permutation importance**: the

Feature	Importance
relationship	0.4252
capital_gain	0.2181
education_num	0.2042
capital_loss	0.0690
age	0.0379

Table 5.15: Feature Importances

model's performance metric, such as accuracy or mean squared error, is calculated before and after randomly shuffling the values of each feature one at a time. The difference in the performance metric before and after shuffling indicates the importance of that feature. A larger drop in performance suggests that the feature is more important to the model's predictions. Remarkably, this is a model-agnostic method, in that no assumptions are made about the model, and we only take its predictions into account when shuffling the data. Figure 5.5 confirms what we have already seen above: relationship, capital_gain and education_num are the most important features for the model.

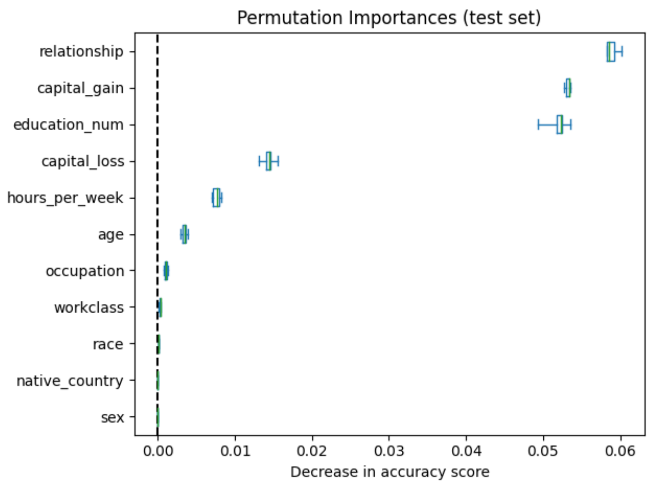


Figure 5.5: Permutation importance

The last methodology employed is another model agnostic one: **partial dependence plot**, used to visualize the relationship between the values of a feature and the predicted outcome of the model, while keeping other features constant. It's extremely useful to check how a model behave considering an important feature, thus checking if we encounter a discriminatory behaviour. Figure 5.6 shows that for the values 0 (corresponding to husband) and 5 (corresponding to wife) the dependency is close to 0.5, so the values don't make much difference to the prediction of the model. But for the remaining values the dependence is close to 0, so people in situations such as divorced, widowed or never married are more likely to be classified as earning less than 50,000 a year for this reason.

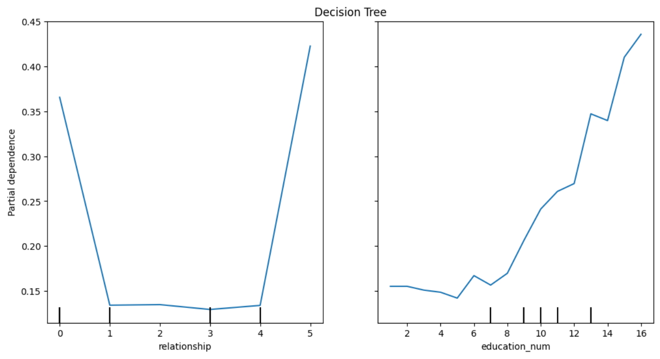


Figure 5.6: Partial Dependence for relationship and education_num

As a final test i provided to the model some samples, where for some features i assigned values associated to people that makes more than 50k a year (Caucasian, 50 years old, American, with a degree and working a full time job) and changed some other features to see how much the prediction changes based on the relationship values. It's easily noticeable from Table 5.16 that there is a correlation between relationship values and prediction, discriminating against people that are not husband nor wife.

Relationship	**Gender**	**Prediction**
Husband	Male	1
Wife	Female	1
Not-in-family	Male	0
Not-in-family	Female	0
Other-relative	Male	0
Other-relative	Female	0
Own-child	Male	0
Own-child	Female	0
Unmarried	Male	0
Unmarried	Female	0

Table 5.16: Prediction Results

Given the situation, we can now add synthetic data using CTAB-GAN+. Due to the situation requiring the employment of oversampling, differential privacy won't be used. Still the generation process of CTAB-GAN+ has been modified, so that the conditional vector can now be used to specify to the model the values for a subset of attributes. In this way i was able to generate samples with specif values for both relationship and income_class. The main idea was to add more samples with income_class equal to 1 where there was an unbalanced ratio for specific values. Also i added some samples to classes that were balanced but a minority in the group, like Other_relative.

We start start by looking at the statistics before and after adding the synthetic data. In Table 5.17 we can observe that the discrimination is already present in the training data: all values except 'Husband' are under-represented,and, except 'Wife', are also unbalanced with respect to the labels. Considering the above results, it seems like the problem is not in the distribution of values but in the distribution of the labels for each value. With this idea in mind i added the synthetic data with the objective of balancing the distribution of labels in the training data, that also influenced the distribution of values for the feature.

	Before		After	
Relationship	Distribution	Label >50k	Distribution	Label >50k
Husband	0.40	0.46	0.28	0.46
Not-in-family	0.26	0.11	0.28	0.43
Other-relative	0.03	0.04	0.09	0.51
Own-child	0.14	0.02	0.15	0.39
Unmarried	0.11	0.07	0.16	0.54
Wife	0.05	0.48	0.04	0.48

Table 5.17: Distribution of values and labels before and after adding synthetic data

By analyzing the new feature importance for the new classifier in Table 5.18. We can see that the model still contains the feature relationship but its importance has dropped from 0.32 to 0.07. Now the models takes into consideration feature there are actually important for classifying the annual earning of an individual like his education level, the capital gain and how many hours he works.

The permutation feature importance (Figure 5.7) confirms that the relationship importance has dropped compared to other features.

As we have done before, we can use partial dependence plot for measuring the influence of each value in the decision process. Compared to before, now each value is more balanced compared to the other Figure 5.8, and the experiment

Feature	Importance
education_num	0.3040
capital_gain	0.1962
hours_per_week	0.1763
relationship	0.0774
age	0.0705

Table 5.18: Feature Importances for balanced dataset

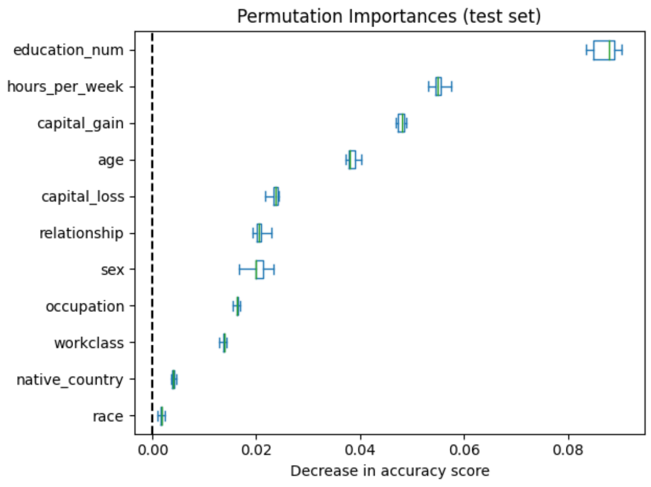


Figure 5.7: Permutation importance for balanced dataset

of modifying only the value of relationship and not the others now yield more balanced results, as seen in Table 5.19.
We can see that there is still some difference in the results, but this is due to gender: the dataset is also unbalanced in terms of the gender of the person and the label. This part will be better analysed in the next section.

The last thing worth mentioning is the accuracy: by adding this synthetic data and changing the hyper parameters the accuracy dropped from 85% to 80%. This is probably due to the fact that the test set remained unbalanced, and now that the model is not using relationship for classifying the samples it has more difficulties. On the other hand if we analyze the confusion matrix with respect to the relationship values we have we can see that now the False Positive Rate is a bit higher but the False Negative Rate diminished: this is implying that now the classifier is not assigning based on the relationship value but considering other situation that better model the task. In Figure 5.9 we can see a comparison of the confusion matrix for sample with value Other_relative, before and after adding the synthetic data.

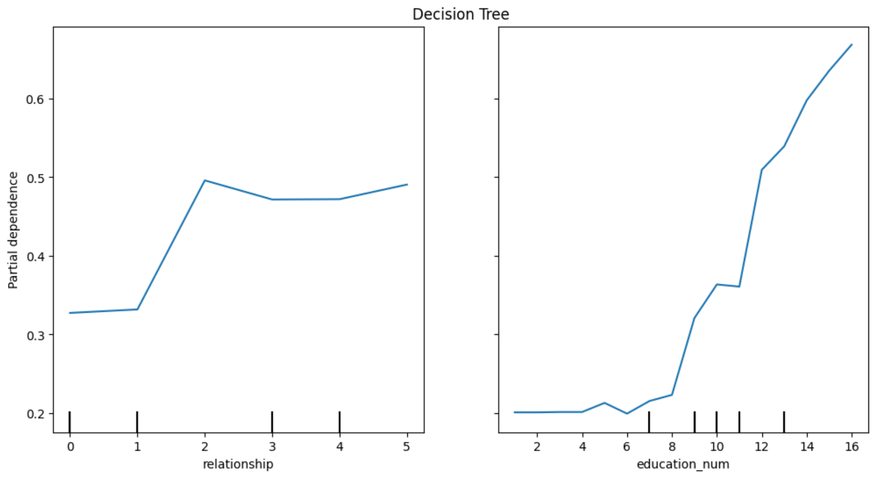
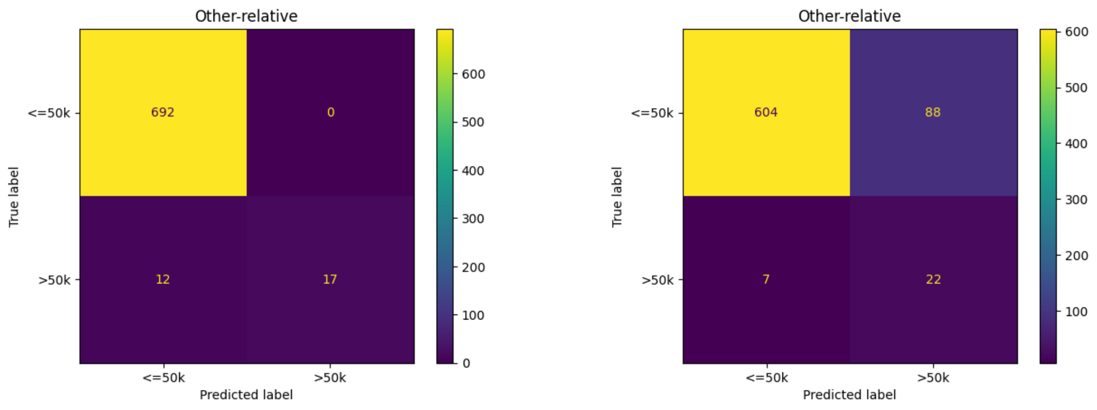


Figure 5.8: Partial Dependence for relationship and education_num balanced

Relationship	**Gender**	**Prediction**
Husband	Male	1
Wife	Female	1
Not-in-family	Male	1
Not-in-family	Female	0
Other-relative	Male	1
Other-relative	Female	0
Own-child	Male	1
Own-child	Female	0
Unmarried	Male	1
Unmarried	Female	1

Table 5.19: Prediction Results



(a) Before (b) After

Figure 5.9: Confusion matrix before and after synthetic data

5.3.2 Fairness measures

In machine learning, a given algorithm is said to be fair, or to have fairness, if its results are independent of given variables , especially those considered sensitive. We will use A for referring to the sensitive attributes and R the classifier's prediction. Fairness measures are properties of the joint distribution of the score, sensitive attribute, and the target variable. This properties can be computed given samples from the joint distribution, and are subject to statistical sampling error [6].

The scores that i will be using are:

- **Independence**: the prediction R and the sensitive attribute A are statistically independent. Can be formally expressed as:

$$P(R = 1|A = a) = P(R = 1|A = b) = ... \tag{5.3}$$

  If this holds we will say that the **acceptance rate** for A is the same.

- **Separation**: for this metric the prediction R and the sensitive attribute A are statistically independent given the target value Y. This is computed for both **True positive** and **False positive** rate, formally:

$$P(R = 1|A = a, Y = 1) = P(R = 1|A = b, Y = 1) = ... \tag{5.4}$$

  And

$$P(R = 1|A = a, Y = 0) = P(R = 1|A = b, Y = 0) = ... \tag{5.5}$$

  This is also know as **equalized odds**.

Given this two metrics we will compute the unfairness score: for the Independence we will compute the difference of probability for a binary A or the average difference (Diff Ind), while for Separation we will compute the difference for both TPR and FPR (Diff TPR and Diff FPR).

In this scenario we are not interested in understanding how a model internally works, but how its predictions have an effect on sensitive attributes. This is specially true if we consider **proxy discrimination**: when classifiers do not considers sensitive attributes for making decisions, but the non-sensitive attributes used works as proxies for these attributes [26]. These behaviours may not noticed by looking at the explanations for the models, but need these scores to be measured. I will use as attributes relationship, race and sex. These categorical features have a low cardinality and can be easily used to measures this statistics. For each one i computed the average and the maximum difference between all the values. The dataset used is Adult, without the synthetic data added in the above section.

The results are shown in Table 5.20, where we can see a confirmation of what seen in the previous section: since with the dataset without synthetic samples the model uses the relationship feature for assigning the label, the average difference is higher for both Independence and TPR. Also even if Figure 5.5 and Table 5.15 tell us that the decision tree has very few nodes that use sex and race, due to the proxy discrimination there still is an high average difference, with the difference between Asian and Other reaching 0.2867.

	Diff Ind		Diff TPR		Diff FPR	
feature	average	max	average	max	average	max
sex	0.1406	-	0.0359	-	0.0448	-
race	0.0960	0.1858	0.1301	0.2867	0.0389	0.081
relationship	0.2046	0.4082	0.2087	0.4263	0.0984	0.2137

Table 5.20: Unfairness scores

The effect of adding the synthetic data for the relationship feature can be seen in Table 5.21, where after adding the data i recomputed the unfairness scores. Two things immediately catches the eye: not only all the differences for relationship diminished (both average and maximum), but also for race and sex there has been some improvement. The synthetic data contained a more enriched data concerning race and sex that helped the model at diminishing the TPR difference: for example for the 'Unmarried' value the ratio of White people diminished from 81% to 71%, while for . Still we can notice that the FPR for gender increased of 3.5%, probably due to a greater presence of male in the synthetic data (due to the label), and the Independence for relationship is still not perfect (the difference between Wife and Own-child is 0.24).

To conclude this part we have to understand that the limits of this methodology and similar techniques relies on trying to solve a socio-technical problem with a technical solution, which is too limited for getting to an actual solution. As [21] and other authors discussed, Artificial Intelligence relies on a model learning all the pattern and biases of the dataset, which in turns reflect the biases of our

	Diff Ind		Diff TPR		Diff FPR	
feature	average	max	average	max	average	max
sex	0.1591	-	0.0011	-	0.0796	-
race	0.0867	0.1469	0.0624	0.122	0.0489	0.0785
relationship	0.1213	0.2402	0.0825	0.1692	0.0248	0.0654

Table 5.21: Unfairness scores with synthetic data for relationship

society. Since we live in a society that for historical and cultural reasons has developed many biases toward minorities, taking for example a lower percentage of women enrolling in STEM degrees or immigrants earning less than the average salary, our datasets will have to include this biases to be a good representation of our world. But then our models will find this pattern in our data and incorporate those in their decision processes, resulting in biased models, that in turn will creating a feedback loop, in which bias in algorithms can potentially be reinforced over time and exacerbated. Considering that Adult is 30 years old [4], it becomes challenging to train a classifier and even expect it to work on our society, with all shaping that have happened in the last year.

Still, a methodology based on synthetic data was able to diminish the importance of a sensitive attribute, as seen in 5.3.1, while also halving the difference in TPR and FPR. This worsening the accuracy of 5%, since now our training set is not correctly reflecting the society it was built on. Furthering modifying the synthetic data (for example introducing samples representing divorced people owning a child) could diminish even more the difference, but would a produce a model incapable of working in our society.

Chapter 6

Results and Discussions

This chapter will answer the research questions i had propose for each experiments with the results on obtained.

For the **Oversampling** part i was interested in a more clear comparison between traditional methods, like GMM and SMOTE, and methods based on GANs such as CTAB-GAN+ and CTGAN, in order to address *will GANs-based approaches overcome traditional models*? Undoubtedly CTAB-GAN+ and CTGAN are better for dataset with categorical data that represents different patterns, but in my experiment i showed that using a dataset with only continuous features, that still represented non-linear behaviours, the results were comparable. While SMOTE represented a weaker improvement as the ratio of the minority diminished, GMM was able to produce an improvement comparable to the two GANs. Considering the time needed to train a GAN and the necessity of using a TPU, GMM still remains a valid alternative.

For the **Privacy** part i performed the MIA attack on the two GANs and a perturbation based approach, in order to address *are GANs approaches better at withstanding the MIA attack*? While keeping a high statistical similarity with the original data, the synthetic data produce by CTAB-GAN+ and CTGAN was able to withstand the attack bringing it to an accuracy of only 50%. They also proved to be robust when the training data was diminished, showing a good capacity at generalizing and not overfitting the data.

Lastly, in the **Fairness** i address *Will introducing synthetic data in the training set diminish the risk of bias*? How could the synthetic data affect the risk of bias of a model, measured with both XAI techniques and fairness measures? By introducing the data produced by CTAB-GAN+ not only the importance of sensitive features diminished, but also the average difference of probability, TPR and FPR diminished. While this proved that we were able to produce a more fair model without degrading the accuracy, i discussed on how this methodology is limited by being a technical solution to a socio-technical solution.

Chapter 7

Limitations

One of the main limitations of this research work can be found on the use of only two datasets. In future works using a variety of datasets for different tasks can provide more insights to add to my results. It could happen that for other datasets with only continuous features the improvement of GAN-based mechanisms is actually higher compared to other traditional mechanisms, or more datasets produce synthetic data with a lower capacity of withstanding the MIA attack, thus resulting in a more important use of the DP-SGD during the train.

This thesis uses only two GAN model, CTGAN and CTAB-GAN+. This decision was done due to the limit hardware resources in my possession, requiring a longer time to train the models thus bringing a substantial time to compute even one single results. Future works could introduce more GAN based approaches, that may be substantially more different in terms of implementation or strategies to learn the distribution of data.

Chapter 8

Conclusions and Future Works

This thesis provide a systematic literature review of the state-of-the-art of generative mechanism of synthetic data, with a special focus on approaches based on Neural Networks. Adopting the most recent proposal of mechanisms and more traditional mechanisms such as GMM and perturbation-based, i was able to derive a set of results that can bring new insight in the field: while for less complex dataset the improvement of using GAN-based mechanisms become equal to improvement obtained with traditional approaches, considering the withstand of a MIA attack GANs are more robust. Future works could try to improve the capacity of current models for other types of tabular data, given the already good results with tabular data containing categorical data, or produce new mechanism able to obtain similar results with a less demanding training procedure.

Regarding the fairness, by introducing synthetic data i was able to diminish the risk of bias of a model while also keeping an acceptable accuracy. This proves that using synthetic data in a model could be a first step to diminish the risk of bias, while keeping in mind that we are trying to solve a socio-technical problem with just a technical solution. Synthetic data created with the objective of maximizing the fairness will loose the original statistical information, where some of the biases are present, and thus become less useful for an application use. Building AI systems is not only a technical problem but also an ethical one due to their impact on society.

The results of this thesis are promising and provide a starting point for research and development of new state-of-the-art methods for generating synthetic data with an acceptable trade-off between privacy preservation, fairness and accuracy.

Bibliography

[1] Martin Abadi et al. “Deep Learning with Differential Privacy”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS’16. ACM, Oct. 2016. DOI: 10.1145/2976749.2978318. URL: http://dx.doi.org/10.1145/2976749.2978318.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 214–223. URL: https://proceedings.mlr.press/v70/arjovsky17a.html.

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023.

[4] Barry Becker and Ronny Kohavi. *Adult*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20. 1996.

[5] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[6] Alessandro Castelnovo et al. “A clarification of the nuances in the fairness metrics landscape”. In: *Scientific Reports* (2022).

[7] Dingfan Chen et al. “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. ACM, Oct. 2020. DOI: 10.1145/3372297.3417238. URL: http://dx.doi.org/10.1145/3372297.3417238.

[8] Edward Choi et al. “Generating Multi-label Discrete Patient Records using Generative Adversarial Networks”. In: *Proceedings of the 2nd Machine*

Learning for Healthcare Conference. Vol. 68. Proceedings of Machine Learning Research. PMLR, 2017, pp. 286–305. URL: https://proceedings.mlr.press/v68/choi17a.html.

[9] Kate Crawford. *Atlas of AI*. Yale University Press, 2021.

[10] Luca Deck et al. "A Critical Survey on Fairness Benefits of Explainable AI". In: *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '24. Rio de Janeiro, Brazil: Association for Computing Machinery, 2024, pp. 1579–1595. ISBN: 9798400704505. DOI: 10.1145/3630106.3658990. URL: https://doi.org/10.1145/3630106.3658990.

[11] Payal Dhar. "The carbon impact of artificial intelligence". In: *Nature Machine Intelligence* (2020). DOI: https://doi.org/10.1038/s42256-020-0219-9.

[12] Ugo Fiore et al. "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection". In: *Information Sciences* 479 (2019), pp. 448–455. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2017.12.030. URL: https://www.sciencedirect.com/science/article/pii/S0020025517311519.

[13] Joao Fonseca and Fernando Bacao. "Tabular and latent space synthetic data generation: a literature review". In: *Journal of Big Data* 10.115 (2023). DOI: https://doi.org/10.1186/s40537-023-00792-7.

[14] R. Stuart Geiger et al. ""Garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data?" In: *Quantitative Science Studies* 2.3 (2021), pp. 795–827. ISSN: 2641-3337. DOI: 10.1162/qss_a_00144. URL: http://dx.doi.org/10.1162/qss_a_00144.

[15] Sahil Girhepuje. *Identifying and examining machine learning biases on Adult dataset*. 2023. arXiv: 2310.09373 [cs.CY].

[16] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[17] et al Gulrajani Ishaan. "Improved training of wasserstein gans". In: *arXiv* (2017). DOI: arXiv:1704.00028.

[18] Zhanglong Ji, Zachary C. Lipton, and Charles Elkan. "Differential Privacy and Machine Learning: a Survey and Review". In: *arXiv e-prints*, arXiv:1412.7584 (Dec. 2014), arXiv:1412.7584. DOI: 10.48550/arXiv.1412.7584.

[19] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. "The Composition Theorem for Differential Privacy". In: *IEEE Transactions on Information Theory* 63.6 (2017), pp. 4037–4049. DOI: 10.1109/TIT.2017.2685505.

[20] Shiva Prasad Kasiviswanathan et al. "What Can We Learn Privately?" In: *2008 49th Annual IEEE Symposium on Foundations of Computer Science* (2008), pp. 531–540. URL: https://api.semanticscholar.org/CorpusID:1935.

[21] Sartori Laura and Theodorou Andreas. "A sociotechnical perspective for the future of AI: narratives, inequalities, and human control". In: *Ethics and Information Technology* (2022).

[22] Robert J. Lyon et al. "Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach". In: *Monthly Notices of the Royal Astronomical Society* 459 (2016), pp. 1104–1123. URL: https://api.semanticscholar.org/CorpusID:56261429.

[23] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *arXiv* (2014). DOI: arXiv:1411.1784.

[24] N. Nilsson. *The Quest for Artificial Intelligence.* Cambridge University Press, 2010.

[25] Noseong Park et al. "Data Synthesis based on Generative Adversarial Networks". In: *Proc. VLDB Endow.* 11 (2018), pp. 1071–1083. URL: https://api.semanticscholar.org/CorpusID:47017667.

[26] Anya E. R. Prince and Daniel Schwarcz. "Proxy Discrimination in the Age of Artificial Intelligence and Big Data". In: *Iowa Law Review* (2020).

[27] Mukul Rathi. *Deep Learning with Differential Privacy (DP-SGD Explained).* URL: https://mukulrathi.com/privacy-preserving-machine-learning/deep-learning-differential-privacy/.

[28] Lucas Theis, Aäron van den Oord, and Matthias Bethge. "A note on the evaluation of generative models". In: *CoRR* abs/1511.01844 (2015). URL: `https://api.semanticscholar.org/CorpusID:2187805`.

[29] Chundawat VS et al. "Tabsyndex: a universal metric for robust evaluation of synthetic tabular data". In: *arXiv* (2022). DOI: `https://doi.org/10.4855/arXiv.2112.09238`.

[30] Lei Xu and Kalyan Veeramachaneni. "Synthesizing Tabular Data using Generative Adversarial Networks". In: 2018. DOI: `arXiv:1811.11264`.

[31] Lei Xu et al. "Modeling Tabular data using Conditional GAN". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: `https://proceedings.neurips.cc/paper_files/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf`.

[32] Chenhan Zhang et al. "Generative Adversarial Networks: A Survey on Attack and Defense Perspective". In: *ACM Comput. Surv.* (2023). DOI: `10.1145/3615336`. URL: `https://doi.org/10.1145/3615336`.

[33] Hongyi Zhang et al. "mixup: Beyond Empirical Risk Minimization". In: *ArXiv* abs/1710.09412 (2017). URL: `https://api.semanticscholar.org/CorpusID:3162051`.

[34] Zilong Zhao et al. "CTAB-GAN: Effective Table Data Synthesizing". In: *PMLR* (2021).

[35] Zilong Zhao et al. "CTAB-GAN+: enhancing tabular data synthesis". In: *Frontiers in Big Data* 6 (2022). URL: `https://api.semanticscholar.org/CorpusID:247922798`.