UNIVERSITY OF CALIFORNIA

Los Angeles

Privacy Auditing of Tabular Synthetic Data Generators Using Membership Inference Attacks

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Statistics

by

Nicklaus Jun Kim

2023

ABSTRACT OF THE THESIS

Privacy Auditing of Tabular Synthetic Data Generators
Using Membership Inference Attacks

by

Nicklaus Jun Kim
Master of Science in Statistics
University of California, Los Angeles, 2023
Professor Guang Cheng, Chair

Synthetic data is a promising technology with numerous benefits for data sharing and machine learning workflows, such as augmenting available data, bolstering fairness, and implementing privacy. As synthetic data becomes more and more prevalent, understanding its methodology and strengths and weaknesses in regards to the above promises becomes crucial. This thesis in particular explores the privacy implications of generative models for synthetic data by investigating whether these models deliver on their proposed privacy guarantees. We analyze the increasingly popular GAN-based methods for generating synthetic data, including CTGAN, DP-CTGAN, and PATE-GAN. To investigate the privacy delivered by these methods, we focus on the approach of adversarial privacy auditing, which utilizes a toolbox of adversarial attacks to detect privacy leakage in (differentially private) algorithms. We aim to extend previous work in privacy auditing, which typically focuses on general machine learning algorithms, to the scarcely examined case of synthetic data generators by analyzing a range of models and datasets. Our goal is to demonstrate the need for further exploration and application of privacy auditing in the scenario of synthetic data, provide insights by comparing behavior across different datasets, and offer simulation results for future investigations into various privacy preservation patterns. For example, experimental results on various datasets and target data records reveal differences in privacy outcomes, highlighting the important role of the *data*, independent of the synthetic data generator, in privacy preservation. The findings highlight the importance of data-centric privacy evaluations and the need for further work to achieve a truly tight privacy analysis. This research hopes to serve as a foundation for further studying adversarial privacy auditing and possibly for the development of robust defenses against privacy leakage in the future.

To my family who have always supported me: my mother, father, sister, and grandmother.

And to the friends who have laughed and whined with me along the way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Thank you to the Department of Statistics and Data Science at UCLA for giving me opportunities to fund my degree and for helping me through unconventional means to grow and apply my academic potential. There are also several individuals in particular to whom I owe an acknowledgement for their contributions to the successful completion of this research endeavor. First, I would like to express appreciation to my advisor, Professor Guang Cheng, and my other committee members for the guidance and support they provided throughout the duration of this project. I am also grateful to Dr. Chi-Hua Wang, whose mentoring and advice have played a pivotal role in helping shape this paper. Finally, I would like to thank my colleagues in my lab with whom I participated in many valuable discussions. I am fortunate to have spent time in an academic environment with such peers and mentors.

CHAPTER 1

Introduction

Synthetic data is an emerging new technology that brings several potential benefits to data processing and analysis. It is being rapidly developed and adopted into many modern machine learning pipelines which demand vast amounts of data, and some experts even estimate that in a few years' time, synthetic data may actually overshadow real data in terms of volume. Therefore, it is becoming increasingly prudent to develop research on synthetic data methodology and understand its strengths and weaknesses and overall behavior in comparison to the true, raw data. Some of synthetic data's valuable use cases include bolstering data volume for ever-increasing data requirements in deep learning, improving fairness of decision-making systems, dealing with class imbalances for categorical features, and implementing overall data privacy, among others. Privacy in particular has garnered much attention, both by researchers and the general public alike, due to the recent boom in personal data collection and sharing. Much of privacy is dictated by whether we happen to expose any sensitive information about a given individual in a dataset; for example, when handling demographic or healthcare data, it is extremely important to ensure no individual's sensitive details are being leaked. Privacy can be formally defined in many ways, depending on the particular data usage scenario (e.g., querying a database, data release to the public domain, or machine learning model training), but overall, the most commonly used metric for privacy is differential privacy [2]. Differential privacy strives to serve as an ironclad guarantee of the privacy of a given algorithm, promising that the probability of an outcome of a computation or analysis remains nearly the same whether any given individual's data is included or excluded from the dataset. Thus, a malicious attacker can not make any inference or draw any conclusions from that individual's data. The level of differential privacy added by a given mechanism is calibrated based on a privacy parameter called epsilon (ε), which controls the level of privacy protection. A larger ε value imposes looser privacy constraints while a smaller ε indicates stronger privacy protection but may lead to decreased fidelity to the original data or less downstream utility on statistical tasks using the newly privatized data. To achieve differential privacy, various mechanisms and techniques have been developed, including data perturbation, noisy querying, and most recently, *synthetic data*.

While many methods have been developed to solve the privacy problem, the hope is that synthetic data can be a "catch-all" solution to data privacy. The promise of synthetic data is its ability to take any real, raw dataset, capture important properties, and provide a way of repeatedly generating or sampling data of user-specified size which has preserved the properties at hand. Additionally of course, the synthetic data can also serve as a privacy mechanism as well, protecting the vulnerable real data. Simply put, a synthetic generation pipeline will consist of two core pieces: a *fitting* step where we fit the model in question to our raw dataset and a *sampling* step where we are able to continually sample data records which are evidently not real yet, when analyzed on a sample-level, will maintain many of the desired essential characteristics, like privacy.

To this point in its research and development history, synthetic data has been overwhelmingly applied to image and text data, particularly given recent leaps in computer vision and natural language processing. However, the majority of data out there is of the *tabular* form, where data records are organized in a table format with a mix of numeric, categorical, and other feature types. It is therefore prudent to focus efforts on tabular synthetic data to meet the current demands of data in industry and academia alike today. Also, apart from the fact that it is the most prevalent form of individual data, tabular data can arguably be considered the most crucial form of data for which we would like to guarantee privacy. For instance, consider healthcare data such as electronic health records, or EHRs; both of these types of data come in tabular form and contain extremely sensitive information about individuals.

Naturally, one's intuition would be to suppose that since synthetic data is constructed in a way such that none of its records directly represents an actual individual, it is automatically private for all intents and purposes. However, this is not necessarily the case. First, it is crucial to define what is meant by "privacy." Countless metrics have been proposed as measuring sticks for data privacy, each with its own strengths and weaknesses under different use cases, but it is needless to say that no matter what, the question of privacy comes down to a lot more, even if the synthetic data records themselves appear to be "fake" and not corresponding to any real individual. One commonly studied example of this principle is the fact that many generative models have been shown to be prone to copying the original data they are trained on, due to overfitting [15]. Particularly for use cases such as healthcare data, we would like to ensure that the generative model at hand is not simply copying the real, original data [23]. In addition to data copying, there may also exist many other possible breaches to a dataset's or an individual's "privacy" that are related to questions of statistical inference, estimation, etc. In later sections we will see in further detail how outcomes such as the performance of predictive models may be indicators of violated privacy in synthetic data.

The above discussion serves to show that the numerous proposed privacy metrics may offer different angles on what privacy actually means, whichever privacy metric we use. This leads us to naturally begin to wonder how we actually *use* privacy metrics, such as differential privacy, in the assessment of synthetic data generation models. While some work has been done on evaluating vulnerability of models [18] and even on private machine learning methods, such as gradient clipping [9], to this point the corresponding analysis on synthetic data models is sparse. We would like to further investigate the underlying privacy of generative models for synthetic data in particular – namely, can we confirm (or deny) that a given synthetic data mechanism is actually delivering on its promise of privacy?

The common procedure for evaluating privacy guarantees is referred to as *privacy auditing*. Privacy auditing may be performed on a given algorithm as a means of assessing and validating the actual level of privacy protection provided by a differentially private algorithm for synthetic data. It involves analyzing the data released by a privacy-preserving mechanism to determine whether the mechanism adheres to the desired privacy guarantees. Overall, privacy auditing plays a crucial role in ensuring that a mechanism meets the intended privacy objectives, helping to verify whether the privacy guarantees claimed by the data generating algorithm are effectively implemented and upheld. Privacy auditing can involve various techniques and methodologies. For example, an audit may include analyzing the privacy parameters, such as epsilon (ε), used in differentially private algorithms and checking whether theoretical bounds on ε are being satisfied. Furthermore, privacy auditing may involve conducting statistical tests or simulations to assess the robustness of the privacy protection mechanisms. This can involve measuring the likelihood of re-identification attacks, assessing the accuracy of query results, or evaluating the overall privacy-utility trade-off.

In recent years, alongside work in adversarial machine learning, some have proposed an adversarial approach to privacy auditing. Notably, Stadler et al. [19] suggest a systematic framework for this adversarial auditing process, providing a quantitative evaluation of the privacy gained from synthetic data publishing and a comparison to the privacy preservation of previous techniques for data anonymization. In contrast to previous approaches, which were *model-specific* and relied upon similarity metrics between real and synthetic data, this framework is able to evaluate the privacy risks of synthetic data publishing itself. Specifically, by assessing the expected privacy gained or lost under various adversarial attacks, e.g., membership inference attacks, we can produce comparative studies of privacy risk across different choices of synthetic data generation mechanisms. A recent extension to this was proposed by Houssiau et al. in [7] and the associated framework `TAPAS`, which is available as an open-source Python package. This new framework contains the evaluation framework first proposed by Stadler et al. but additionally offers a large suite of different attacks and other choosable scenarios for performing an adversarial auditing of a given generator. We utilize this framework to produce the results and visualizations in this paper, but it should be noted that there exist many alternative frameworks and packages which offer a very similar toolbox of implementations and assessments of attacks.

We wish to extend upon these previous works as well as others in the privacy auditing literature and provide an in-depth look at this privacy auditing process in practice by applying it to a wider range of datasets and synthetic data generators, particularly those which are commonly used and offer the best performance in other metrics of synthetic data, such as distributional fidelity and predictive performance on downstream machine learning tasks. By doing so, we hope that we can (1) introduce simulation results which may serve as a launchpad for larger-scale experiments and investigation into the true privacy offered by several key synthetic data generators; (2) provide insightful findings by comparing behavior under adversarial attacks on different datasets and, importantly, different data records; (3) demonstrate the need for further exploration and application of privacy auditing when adopting models advertised as being private to a certain degree. Overall, since privacy auditing as a field is still new and undeveloped, creating techniques to assess the vulnerability of differentially private algorithms to various attack scenarios and designing robust defenses against those attacks are important aspects of privacy auditing, and we hope to provide a strong starting basis for researching such topics in more depth.

CHAPTER 2

Background

In this section we give an overview of the core concepts in synthetic data and privacy, some related work, and other preliminary ideas.

2.1 Tabular Data

A tabular dataset D can be said to contain N_c continuous columns $\{C_1, \ldots, C_{N_c}\}$ and N_d discrete columns $\{D_1, \ldots, D_{N_d}\}$, where each column is a random variable. These random variables follow some joint distribution, which we can write as $\mathbb{P}(C_{1:N_c}, D_{1:N_d})$. The columns of the dataset represent the features or variables of our data, and there is some underlying joint data distribution over all of these features. We regard each row or record $r_j, j \in \{1, \ldots, n\}$, as an observation "sampled" from the joint distribution. A related notion is that of a marginal; many popular synthetic data generation algorithms utilize marginals and subsequent conditional distributions in order to generate data.

In the above definition, we have limited our discussion of features in a tabular dataset to only continuous and discrete types. Of course, other variable types are quite prevalent in data, such as character/text and time-series data. For the intents of synthetic data, each of these other data types is typically dealt with in a particular, prescribed way such that we can usually just treat them as being either continuous or discrete. For instance, it is quite common for categorical character columns – e.g., "degree type" being one of "High School", "Bachelor's", "Master's", etc. – to be encoded using a technique such as one-hot encoding.

After such a preprocessing or encoding, the column will essentially be an ordinary discrete column and no longer of a special data type (i.e., text).

2.2 Synthetic Data Generators

When discussing methods for synthesizing data, there are a few competing philosophies for the generation algorithm design; the differences among the approaches essentially come down to the question of how to treat a row of the table. This treatment, in turn, dictates under what logic we may fit a model to real rows and sample *synthetic* rows.

One philosophy for synthetic data generation is that of marginal-based methods, perhaps the most fundamental and straightforward approach. Marginal-based methods generate synthetic data by modeling the marginal distributions of each column independently. For example, one popular approach is histogram-based methods, which involve partitioning the data into bins and estimating the probability density function for each bin. Samples are then generated by randomly selecting a bin, then randomly sampling from the estimated probability density function for that bin. To generate synthetic data, we may then sample from each estimated distribution to create each new row for the tabular dataset. That is, to obtain each synthetic row, we sample the synthetic values for each column individually from the marginals. It is also possible to extend this approach to using not just single column, or one-way, marginals, but also higher-order marginal distributions in order to capture the statistical dependencies and other relationships among columns. For instance, two-way marginals would capture the correlation between two given columns of a tabular dataset [20]. Some popular marginal-based methods include Noisy Histogram [8], IndHist [17], and MST [13], which was adapted from the winning algorithm from the 2018 synthetic data competition hosted by the National Institute of Standards and Technology (NIST)[1].

[1]https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2018-differential-privacy-synthetic

Related to marginal-based methods is the notion of methods based on graphical models. Graph-based methods model the dependencies between variables using a graph structure. Each node X_i in the graph represents a variable, and the edges between nodes represent the dependencies between variables. One popular approach is to use a Bayesian network [6] to represent the graph structure and conditional probability distributions, as in [25] and [14]. Mathematically, a Bayesian network can be represented as a directed acyclic graph (DAG), where each node represents a variable (i.e., column of the dataset), and each directed edge represents a conditional dependence between the two variables. The joint probability distribution over all variables in the network can then be written as

$$\mathbb{P}(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} \mathbb{P}(X_i \mid \mathrm{pa}(X_i)),$$

where X_i is the i-th variable, and $\mathrm{pa}(X_i)$ represents the parents of X_i in the graph. To generate synthetic data, we can sample from the joint distribution over all variables in the network, conditioned on the observed data. This family of methods is rather similar in spirit to the marginal-based methods discussed above but differs in that we are now assuming some knowledge of a structure of conditional independence (or dependence) relations which can be conveniently exploited and captured in a graphical model. Overall, marginal-based methods typically assume that each column is independent, while graph-based methods model the dependencies between variables using a graph structure. The choice of marginal-based versus graph-based methodology will depend on the specific problem and the properties of the data that need to be captured.

Next, in recent years, there has been a proliferation of methods based on generative adversarial networks (GANs) [5]. Of course, in the landscape of generative modeling as a whole, GANs have dominated data generation in computer vision and fostered much research; recently, a number of researchers have adapted GANs to the task of synthetic data generation for tabular data as well. GANs are a popular deep learning architecture, where the basic idea is to train two neural networks simultaneously: a generator network, which produces synthetic data samples, and a discriminator network, which distinguishes between real and synthetic data samples. The generator is trained to produce samples that are indistinguishable from real samples, while the discriminator is trained to correctly classify real and synthetic samples. This leads to a training process which can be described as a two-player minimax game between the generator and the discriminator, where the objective function for the generator is to minimize the probability of the discriminator correctly classifying synthetic samples, while the objective function for the discriminator is to maximize the probability of correctly classifying real and synthetic samples. This can be written mathematically as

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

where $\mathcal{G}$ is the generator network, $\mathcal{D}$ is the discriminator network, $\boldsymbol{x}$ represents a real data sample, $\boldsymbol{z}$ represents a random noise vector, $p_{\text{data}}(\boldsymbol{x})$ is the distribution of real data, and $p_{\boldsymbol{z}}(\boldsymbol{z})$ is the distribution of the noise vector.

During training, the generator takes a noise vector as input and produces a synthetic sample. The discriminator then classifies this sample as real or synthetic, and the error is backpropagated through both networks. The generator is updated to produce samples that fool the discriminator, while the discriminator is updated to better distinguish between real and synthetic samples. Once the GAN has been trained, the generator can be used to produce synthetic data samples. To do this, a noise vector is sampled from the noise distribution and passed through the generator. The output of the generator is a synthetic sample that can be used as if it were a real sample. Methods utilizing GANs specifically for the *tabular* synthetic data generation task offer improvements for tabular data in particular; we go into further detail on such methods below.

Other categories of synthetic data generators exist and continue to be proposed, such as methods based on variational autoencoders (VAEs) like TVAE [21] and diffusion-based methods like STaSy [11]. Since we do not cover any of these methods in this paper, we omit further elaboration on these models but just wanted to mention them here for completeness.

In summary, there exist several candidates for classes of generators, falling into a few different schools of thought regarding how to treat a row of the data, for example:

- Marginal-based: A table row is a series of low(er)-dimensional marginal distributions/an observation of a probabilistic graphical model.
- Graph-based: A table row is an observation of a probabilistic graphical model.
- GAN-based: A table row is an encoded representation vector.

In this paper, we will focus our attention on GAN-based methods, due to the fact that they are at the forefront of tabular synthetic data generators, both in development and applicability, and they possess the most accessible research in other data domains (vision, text) from which we may draw influence and apply to the tabular case. Specifically, we will be looking at the behavior of the following GAN-based methods: CTGAN, DP-CTGAN, and PATE-GAN.

The conditional tabular GAN, or CTGAN, was first proposed by Xu et al. [21] and adapts the GAN algorithm for the purpose of generating *tabular* data. CTGAN is effective for generating synthetic tabular data in particular because it offers several key advantages in contrast to vanilla GANs: handling categorical data, handling continuous data, conditional generation, robust training.

Tabular data often contains categorical variables, which can be difficult to handle in traditional GANs. CTGAN uses a technique called "embedding" to represent categorical variables as continuous values, making them easier to work with in a GAN framework. CTGAN can also handle continuous variables in tabular data, which is a critical feature for generating realistic synthetic data. CTGAN uses a special normalization technique called "min-max normalization" to ensure that the continuous variables in the synthetic data have similar ranges to the original data. The min-max normalization technique scales the continuous variables in the original data to the range [0,1]. Let x_i be a continuous variable in the original data, $x_{\min}$ and $x_{\max}$ be the minimum and maximum values of x_i, respectively. Then the min-max normalization of x_i is given by

$$\hat{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}.$$

The normalized value $\hat{x}_i$ now lies in the range [0,1], which makes it easier to work with in a GAN framework.

Tabular data often has underlying dependencies between variables, and CTGAN can model these dependencies using a conditional GAN architecture. The ability to generate data based on specific conditions is particularly useful in many applications where specific data distributions need to be preserved. In conditional GANs, the generator $\mathcal{G}$ takes both random noise z and a set of conditioning variables y as inputs, and generates synthetic data $\hat{x} = \mathcal{G}(z, y)$. The conditioning variables provide additional information about the data distribution that the generator can use to generate more realistic synthetic data. In CTGAN, the conditioning variables are used to specify the properties of the data that need to be preserved in the generated samples.

Lastly, CTGAN uses a novel loss function that includes a "hinge" loss term, which helps stabilize the training of the GAN. This loss function helps to prevent the generator from producing unrealistic, "unfaithful" data samples, which is a common problem in many GAN applications. The hinge loss is a loss function that penalizes the generator if it generates data that is distributionally too far from the original data. In CTGAN, the hinge loss is defined as

$$\mathcal{L}_{\text{Hinge}} = \max(0, m - \mathcal{D}(x_{\text{real}}) + \mathcal{D}(\mathcal{G}(z, y))),$$

where $\mathcal{D}$ is the discriminator, x_{real} is a real data sample, $\mathcal{G}$ is the generator, z is random noise, y is the conditioning variable, and m is a hyperparameter that controls the margin of the hinge loss. The hinge loss ensures that the generator produces data that is close to the original data distribution and is particularly effective in stabilizing the training of GANs.

The CTGAN synthetic data generator as described in detail above can be extended to a generator which incorporates differential privacy. (We discuss differential privacy in further detail in Section 2.3.) First proposed by Fang et al. [4] for application to medical data in particular, the DP-CTGAN achieves differential privacy by clipping the training gradient, which bounds the gradient norms, and injecting noise. The injection of noise can be carefully calibrated using a variety of methods – there exist numerous mechanisms for noise addition, each with rigorous differential privacy guarantees. Some popular mechanisms are the Laplace mechanism and Gaussian mechanism [3]. To allocate the privacy budget, DP-CTGAN makes use of the privacy accountant to keep track of privacy loss.

Following the success of the CTGAN, DP-CTGAN, and other similar models, Yoon et al. [24] proposed an alternative adaptation of GANs for synthesizing tabular data in the PATE-GAN. PATE-GAN involves the use of the PATE, or Private Aggregation of Teacher Ensembles, framework [16]. Specifically, in the PATE-GAN, the typical GAN discriminator is replaced by a PATE mechanism so that the discriminator is differentially private. We omit further details of DP-CTGAN and PATE-GAN here for brevity.

2.3 Differential Privacy

The job of a synthetic data generator $\mathcal{G}$ is to approximate/estimate the aforementioned joint probability distribution $\mathbb{P}$ over the continuous and discrete columns of a dataset D. A synthetic data generator is a function that takes as input a real dataset $D^{(r)} \in \mathcal{D}$ and outputs a synthetic dataset that satisfies certain requirements such as preserved column similarity, high utility, etc. Formally, we may write a synthetic data generator as a *randomized function* $\mathcal{G} : \Omega \times \mathcal{D} \mapsto \mathcal{D}$, which takes as input a real dataset R and outputs a synthetic dataset $\mathcal{G}(R) = S$. Here, the set Ω denotes a probability measure space, where typically random variables are defined as functions of Ω – we can view this as the ”seed” of the randomized mechanism.

Of course, given a generator, we are very much interested in somehow quantifying its guaranteed or delivered privacy level in its generated synthetic samples. This brings us to the natural extension of desiring *differentially private* synthetic data generators, which are designed to come with the guarantee of differential privacy. The current "gold standard" for measuring privacy is differential privacy. Differential privacy is applicable to any randomized mechanism or algorithm $\mathcal{M}$ (including, for example, a synthetic data generator $\mathcal{G}$ as seen above).

Theorem 1 *Let D, D' be two neighboring datasets differing in exactly one entry. A randomized algorithm $\mathcal{M} : \Omega \times \mathcal{D} \mapsto \mathcal{S}$ is (ε, δ)-differentially private if for all $S \subseteq \text{Range}(\mathcal{M}) = \mathcal{S}$ and all D, D':*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(e)\Pr[\mathcal{M}(D') \in S] + \delta.$$

If $\delta = 0$, we say that the algorithm $\mathcal{M}$ is ε-differentially private. It is somewhat common to see simply ε-differential privacy in many applications and theoretical studies. Differential privacy has been shown to possess several useful corollaries which are particularly valuable when discussing synthetic data. We include the most relevant results, the post-processing theorem [3] and sequential composition theorem [10], here.

Corollary 1 *(Post-processing) Let $\mathcal{M} : \Omega \times \mathcal{D} \mapsto \mathcal{S}$ be a randomized algorithm that is (ε, δ)-differentially private. Let $f : \mathcal{S} \mapsto \mathcal{S}'$ be an arbitrary randomized function. Then $f \circ \mathcal{M} : \Omega \times \mathcal{D} \mapsto \mathcal{S}'$ is also (ε, δ)-differentially private.*

From the above, together with the original definition of (ε, δ)-differential privacy, we can derive the fact that $(\varepsilon, 0)$-differential privacy – or simply ε-differential privacy – composes in a straightforward way, so that composing two ε-differentially private mechanisms results in a mechanism that is (2ε)-differentially private. Formally, we can see that this behavior carries out in the general sense, so that we may "compose" the ε and δ parameters of differential privacy when composing more than one mechanism.

Corollary 2 *(Sequential Composition) Suppose we have k (ε, δ)-differentially private mechanisms $\mathcal{M}_1, \ldots, \mathcal{M}_k$. Then, the composition of these k mechanisms, where each $\mathcal{M}_i$ is $(\varepsilon_i, \delta_i)$-differentially private, yields a mechanism that is $(\sum_i \varepsilon_i, \sum_i \delta_i)$-differentially private.*

Lastly, again following from the above corollary in conjunction with our formal definition of differential privacy, we can define a notion called group privacy.

Corollary 3 *(Group Privacy) Any $(\varepsilon, 0)$-differentially private mechanism $\mathcal{M}$ is $(k\varepsilon, 0)$-differentially private for groups of size k. That is, for all datasets D, D' differing in k entries (cf. D, D' differing in one entry in differential privacy) and all $S \subseteq \text{Range}(]\mathcal{M})$,*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(k\varepsilon)\Pr[\mathcal{M}(D') \in S] + \delta.$$

In a nutshell, the above result states that the strength of the privacy guarantee level decreases linearity with the size or cardinality of the group in question.

Usually when the differential privacy of a given algorithm is discussed or analyzed, the associated guarantee of privacy is very arbitrary and unproven. In other words, an algorithm is usually proposed to be differentially private at a certain level of ε (and δ), with no theoretical results to underpin that promise. Of course, some well-studied algorithms do have proven results regarding privacy, but these results typically come well after the initial proposal of the method and are always method-specific results. This may lead to many such algorithms not being truly, say, ε-differentially private as advertised, due to a bevy of potential issues either in theory or in the available implementation of the method. How, then, can we quantify and/or confirm the true level of differential privacy, or of ”privacy” in general?

CHAPTER 3

Methodology

In order to perform a rigorous privacy audit, we make use of the adversarial framework as in [7, 19]. Namely, we utilize a quantitative framework for assessing privacy gain – or conversely, leakage – using an adversarial attacking toolbox. In this setting, we will use the following notation: the real dataset R, the synthetic dataset S, the synthetic data generator $\mathcal{G}$ with $\mathcal{G}(R) = S$, a target record $\mathbf{r}_t$, an adversary $\mathcal{A}$, and the adversarial advantage $Adv^{\mathcal{A}}$. We suppose that the adversary $\mathcal{A}$ is given S and aims to use it to infer sensitive information about the real individual $\mathbf{r}_t$. By simulating attacks by the adversary $\mathcal{A}$ on a selected target record $\mathbf{r}_t$, we can capture $Adv^{\mathcal{A}}$, i.e., how much including $\mathbf{r}_t$ in the published data increases this individual's privacy risk. We can then define our notion of privacy gain as the added benefit of publishing synthetic dataset S in place of the real one R for the given target record $\mathbf{r}_t$. Formally, we consider the difference between the adversarial advantage with access to S versus R:

$$PG := Adv^{\mathcal{A}}(R, \mathbf{r}_t) - Adv^{\mathcal{A}}(S, \mathbf{r}_t).$$

A high value of PG indicates that publishing S in place of R substantially reduces the privacy risk for a given target $\mathbf{r}_t$. We reiterate that this measurement is done on a by-record basis and depends on the choice of $\mathbf{r}_t$. This is beneficial in that it allows us to assess privacy in the worst-case scenario versus just the "average-case," which is more or less what metrics like differential privacy do overall. That is, we can now investigate privacy on a per-individual level, rather than just at the sample level, so we can find some $\mathbf{r}_t$ which gives better or worse PG than others and therefore find a worst case privacy measurement for some choice of $\mathbf{r}_t$. As a result, we may investigate phenomena such as disparate privacy gain [12], among others.

Next, we must formally define this adversarial advantage $Adv^{\mathcal{A}}$ used in the calculation of privacy gain above. In order to do so, we first need to choose an attack scheme, which we introduce in the following.

3.1 Linkability and Membership Inference

A linkage attack (also called linkability) refers to the ability to link together two or more target records belonging to the same individual from different datasets. In essence, linkage attacks aim to determine whether a target record $\mathbf{r}_t$ is present in some sensitive dataset, in our case the real dataset R. This may enable adversaries to attach an identity to a record that has supposedly been de-identified in the synthetic dataset S.

We can model the risk of linkability as a membership privacy game between a data holder and an adversary $\mathcal{A}$, as in [18].[1] In the context of synthetic data, we formalize linkability as an instance of the membership inference attack in the following way. Suppose we have the real data R of n records, drawn from a population $\mathcal{R}$ with real data distribution $\mathcal{D}_{\mathcal{R}}$, and we sample synthetic data S from a SDG trained on R: $\mathcal{G}(R) = S$. The adversary wants to infer whether some given $\mathbf{r}_t \in R$ using only S. Then, we can outline the membership inference game in Algorithm 1.

[1]As a note, linkability cannot be *fully* measured through membership inference; linkability is actually a stronger privacy breach. In fact, establishing a successful linkage between a target $\mathbf{r}_t$ and some records in a dataset R implies membership of $\mathbf{r}_t$ in R. However, the opposite is not true: even if membership of $\mathbf{r}_t$ in R is established, the attacker still does not know which records of R are directly linked to $\mathbf{r}_t$.

Algorithm 1 Linkability/membership inference game

```
Require: Target record $\mathbf{r}_t \in \mathcal{R}$
  Sample real data $R \sim \mathcal{D}_{\mathcal{R}}^{n-1}$.
  Draw $s_t \sim \{0, 1\}$.
  if $s_t = 0$ then
    Add random record $\mathbf{r}_r \sim \mathcal{D}_{\mathcal{R}\setminus\mathbf{r}_t}$.
    $R \leftarrow R \cup \mathbf{r}_r$.
  else if $s_t = 1$ then
    Add target record $R \leftarrow R \cup \mathbf{r}_t$.
  end if
  Fit generator $\mathcal{G}$ on $R$.
  Sample synthetic data $S \sim \mathcal{D}_{\mathcal{G}(R)}^{m}$.
  Assign $X \leftarrow S$.
  Make a guess $\widehat{s_t} \leftarrow \mathcal{A}^{\mathcal{L}}(X, \mathbf{r}_t, \mathcal{P})$.
```

Now that we are equipped with the procedure for carrying out membership inference, we can define the adversary's advantage specific to this particular case of linkability. As in [22], we may write the *linkage* adversary's advantage as

$$\begin{aligned} Adv^{\mathcal{L}}(X, \mathbf{r}_t) &:= 2\mathbb{P}\left[\mathcal{A}^{\mathcal{L}}(X, \mathbf{r}_t, \mathcal{P}) = s_t\right] - 1 \\ &= \mathbb{P}\left[\widehat{s_t} = 1 | s_t = 1\right] - \mathbb{P}\left[\widehat{s_t} = 1 | s_t = 0\right], \end{aligned}$$

where X may represent a real or synthetic dataset. For each target $\mathbf{r}_t$, the adversary will instantiate the game multiple times and this (linkage) adversary advantage can be measured, conditioned on the challenger's choice of X. If the full raw dataset R is published, then the adversary can simply check whether $\mathbf{r}_t \in R$ and its "guess" will be totally deterministic; that is, $\mathcal{A}^{\mathcal{L}}(R, \mathbf{r}_t) = 1$. If instead the synthesized dataset S is published, then the adversary must perform a sort of binary classification on whether $\mathbf{r}_t \in R$, using the features of S. Using this classification paradigm, we can consider the above expression for $Adv^{\mathcal{L}}(X, \mathbf{r}_t)$ as the difference between the true positive rate and the false positive rate. We cover the details on how to actually train the classifier in Section 3.3.

We can also write the *privacy gain* of publishing the synthetic data S in place of the raw data R with respect to the particular risk of linkability:

$$PG = Adv^{\mathcal{A}}(R, \mathbf{r}_t) - Adv^{\mathcal{A}}(S, \mathbf{r}_t) = 1 - Adv^{\mathcal{L}}(S, \mathbf{r}_t).$$

A privacy gain of $PG = 0$ indicates that the adversary infers the presence of the target $\mathbf{r}_t$ in R with perfect accuracy regardless of whether they are given access to R or S. If on the other hand, observing S gives the adversary no advantage in inferring the target's presence ($Adv^{\mathcal{L}}(S, \mathbf{r}_t) = 0$), then $PG = 1$. In [22], the authors also propose a bound on the expected advantage of the adversary, which we can then extend to the specific case of linkability:

$$Adv^{\mathcal{L}} \leq e^{\varepsilon} - 1 \iff PG \geq 1 - (e^{\varepsilon} - 1) = 2 - e^{\varepsilon}.$$

3.1.1 Effective Epsilon

Much like privacy gain, effective epsilon is another useful metric that may be used to measure or quantify privacy under an adversarial attacking framework using membership inference attacks [9]. Effective epsilon, denoted ε_{eff}, is essentially used to evaluate and compare the *actual* privacy protection level achieved by an ε-differentially private algorithm. As with privacy gain, the ε_{eff} metric assesses per-record privacy and not just sample-wide privacy; that is, we can now use ε_{eff} to quantify privacy by individual records instead of only evaluating a single ε value for the entire dataset. Typically, however, as we shall see, effective epsilon is calculated (much like ordinary ε in differential privacy) by taking a worst-case approach across different scenarios. Effective epsilon can be calculated using various methods, such as the moments accountant or the advanced composition theorem, depending on the specific algorithm and scenario at hand. We refer the reader to [9] for further details on the computation process for ε_{eff} upon which we rely in this paper.

To derive an expression for ε_{eff}, we first recall that the adversary wants to classify whether

$\mathbf{r}_t \in R$. The adversary is looking to solve this classification problem and therefore, there exist some associated classification accuracy metrics. We make use of these classification metrics in order to define ε_{eff}. Specifically, there exists a trade-off between an attacker's true positive rate and false positive rate, a relationship which applies even in the worst-case attack model of exact data knowledge [10]. Then, Theorem 2 below gives us a lower bound on the value of ε under the given attack scenario.

Theorem 2 *Let $\mathcal{M} : \Omega \times \mathcal{D} \mapsto \mathcal{O}$ be a randomized mechanism satisfying (ε, δ)-differential privacy and $d, d' \in \mathcal{D}$ be neighboring datasets ($d \overset{\sim}{=} d'$). Then, for any randomized attacker $\mathcal{A} : \Omega \times \mathcal{O} \mapsto \{d, d'\}$,*

$$e^{\varepsilon} \geq \max\left(\frac{TP_{\mathcal{A}} - \delta}{FP_{\mathcal{A}}}, \frac{1 - FP_{\mathcal{A}} - \delta}{1 - TP_{\mathcal{A}}}\right),$$

where $TP_{\mathcal{A}} = \mathbb{P}[\mathcal{A}(\mathcal{M}(d)) = d]$ and $FP_{\mathcal{A}} = \mathbb{P}[\mathcal{A}(\mathcal{M}(d')) = d]$ are true and false positive rates, resp.

In general, the inequality in Theorem 2 is not tight; next, we can define ε_{eff} by taking the upper bound over all attackers, so that

$$\varepsilon_{eff}(\delta; d, d') := \log \sup_{\mathcal{A}:\Omega\times\mathcal{O}\mapsto\{0,1\}} \max\left(\frac{TP_{\mathcal{A}} - \delta}{FP_{\mathcal{A}}}, \frac{1 - FP_{\mathcal{A}} - \delta}{1 - TP_{\mathcal{A}}}\right).$$

ε_{eff} can be used to prove that a privacy analysis is tight and can measure privacy protection in different contexts, i.e., under different assumptions about the attacker $\mathcal{A}$. Specifically, many synthetic data generators $\mathcal{G}$ are made differentially private with respect to their parameters θ, usually through the adding of noise during training. As sampling the model is a form of post-processing, $S = \mathcal{G}_\theta(R)$ is equally (ε, δ)-differentially private, but this guarantee is likely not tight, as sampling introduces additional randomness. Thus, ε^{eff} can help to quantitatively evaluate more realistic privacy guarantees. For instance, one straightforward use case of ε_{eff} would be to compare it to the promised ε level of different generators – or more generally, any ε-differentially private algorithm.

3.2 Threat Modeling Framework

In our adversarial evaluation of privacy, we are working with a wide variety of assumptions and scenarios for our analysis. To ensure that the evaluation is done in a way that is *contextual*, we can formalize these assumptions by using the *threat model framework*. A threat model, or attack model, lays out the setup in which our attacks are performed and the knowledge which we assume the adversary holds. Under a given threat model, an attacker must then learn its "guess" function, i.e., $\mathcal{A}^{\mathcal{L}}$. The threat model we use, as in [7], is modular and consists of three parts which can be modified independently of one another: knowledge of the dataset R, knowledge of the synthetic data generator $\mathcal{G}$, and the attacker's goal. We now outline each of these and their various possible settings.

3.2.1 Data Knowledge

Recall that the adversary is always in full possession of the full synthetic dataset S and is trying to guess whether the target $\mathbf{r}_t$ was used to fit the generator $\mathcal{G}$ that output S. Moreover, the adversary may hold additional, possibly uncertain knowledge about the real dataset R. This leads us to consider two types of knowledge the adversary may have on the original dataset R. In auxiliary data knowledge, the adversary has access to an auxiliary dataset that is row subsampled from R; the remaining unsampled rows make up the testing set for evaluating a trained attack. Using the auxiliary dataset, the adversary is able to generate datasets from the same distribution as R. This is the setup found in [18, 19] as well as other similar works. By contrast, in exact data knowledge, the adversary knows that R is one of two datasets differing in exactly one entry; the adversary knows the entire dataset except the membership of the target $\mathbf{r}_t$. Exact data knowledge is a "worst-case" attack model and can be used to audit privacy in the context of differential privacy due to its worst-case nature, however it is not a very realistic assumption of the adversary.

3.2.2 Generator Knowledge

There exist several possible options for the amount of knowledge the adversary has on the synthetic data generator $\mathcal{G}$:

- White-box: The attacker has exact knowledge of $\mathcal{G}$ and has access to trained parameters of the model θ, e.g., weight and bias terms of a neural network, of $\mathcal{G}$.
- Black-box: The attacker has exact knowledge of $\mathcal{G}$ and can call $\mathcal{G}$ on arbitrary datasets but has no information on parameters θ.
- No-box: The attacker has no knowledge about the generator.

3.2.3 Attacker Goal

The last piece of the threat model is the attacker goal. As mentioned in a previous section, our attack goal in this work is that of linkability, or more specifically, membership inference. In a general sense, membership inference attacks (MIAs) are a class of adversarial attacks that aim at inferring whether a target record is in some dataset ($\mathbf{r}_t \in R$). Another common attacker goal is that of attribute inference attacks (AIAs), in which the adversary, for a target attribute a and incomplete target record $\mathbf{r}_t$ which is missing a, seeks to infer the value of a. Lastly, a less common type of attack is the reconstruction attack, where, as the name suggests, the adversary aims to reconstruct the entire real dataset R. These attacks are not very deeply explored at the moment, while membership inference and attribute attacks are commonly investigated in the privacy literature regarding synthetic data.

3.3 Shadow Modeling Attack

Finally, now that we have considered various metrics for success in privacy auditing and set up our threat model, we must somehow model a way for the adversary to make its classification "guess" $\mathcal{A}^{\mathcal{L}}(X, \mathbf{r}_t, \mathcal{P})$. Specifically, in our case of membership inference, formulating this guess function amounts to guessing whether the target record $\mathbf{r}_t$ is in the real dataset R, as we have previously mentioned. Also, the adversary has black-box access to the generator $\mathcal{G}$ training algorithm and a training dataset $R \sim \mathcal{D}_{\mathcal{R}}^{n}$ (or possibly a reference dataset from an auxiliary distribution $R_{\mathcal{A}} \sim \mathcal{D}_{\mathcal{R}}^{l}$).

Given $\mathbf{r}_t$, the adversary can learn the guess function $\mathcal{A}^{\mathcal{L}}$ as shown in Algorithm 2, according to the formulation in [19]. Note the similarities between this procedure and that of Algorithm 1.

Algorithm 2 Shadow Modeling Attack

```
Require: Target record $\mathbf{r}_t \in \mathcal{R}$
  Sample multiple training sets $R_i$ of size $n$ from the reference dataset $R_{\mathcal{A}}$.
  for $R_i$ do
      Train a generator $g(R_i)$ using the same training procedure as $\mathcal{G}$.
      Sample multiple synthetic datasets $S$ of size $m$ from $g(R_i)$.
      Assign the generated datasets the label $s = 0$.
  end for
  for $R_i$ do
      $R_i' \leftarrow R_i \cup \mathbf{r}_t$.
      Train a generator $g(R_i)$ using the same training procedure as $\mathcal{G}$.
      Sample multiple synthetic datasets $S$ of size $m$ from $g(R_i')$.
      Assign the generated datasets the label $s = 1$.
  end for
  Train a classifier $\mathcal{A}^{\mathcal{L}}$ on the labeled synthetic datasets.
  Output a guess $\widehat{s}_t \leftarrow \mathcal{A}^{\mathcal{L}}(S, \mathbf{r}_t, \mathcal{P})$.
```

Mounting a successful black-box membership attack (e.g. shadow modeling attack) on a generative model $\mathcal{G}$ is very challenging because the adversary needs to identify the influence a single record $\mathbf{r}_t$ has on the high-dimensional data distribution $\mathcal{D}_{g(R)}$ as opposed to a low-dimensional confidence vector. That is, the adversary essentially wants to distinguish between two distributions, $\mathcal{D}_{R\cup\mathbf{r}_r}$ and $\mathcal{D}_{R\cup\mathbf{r}_t}$, using just a single observation $S \sim \mathcal{D}^m_{g(R)}$. One proven way to tackle this high-dimensional classification problem is to use a classifier based on a feature set.

3.3.1 Feature Set Classifier

We have seen that the attack's guess function $\mathcal{A}^{\mathcal{L}}(S, \mathbf{r}_t, \mathcal{P})$ amounts to training a classifier over sets, namely the sets $\mathcal{D}_{R\cup\mathbf{r}_r}$ and $\mathcal{D}_{R\cup\mathbf{r}_t}$. However, this is a very high-dimensional problem and one that is problematic to solve directly. To reduce the effects of high dimensionality and sampling uncertainty on this classification problem, the adversary can apply feature extraction techniques. That is, instead of training a classifier $\mathcal{D}_{R\cup\mathbf{r}_r}$ and $\mathcal{D}_{R\cup\mathbf{r}_t}$ directly on S, we can distinguish feature vectors extracted from those datasets synthesized by generators trained with versus without the target, $\mathcal{D}_{R\cup\mathbf{r}_r}$ and $\mathcal{D}_{R\cup\mathbf{r}_t}$.

A feature set is a function $f(X) = \mathbf{F}$ that takes as input a dataset X from the high-dimensional data domain and outputs a numerical vector $\mathbf{F}$ that maps X into a lower-dimensional feature space. For our purposes, the success of an attack on a synthetic data generator using feature set $\mathbf{F}$ depends on two factors: (1) whether the presence of the target $\mathbf{r}_t$ has a detectable impact on any of the features in $\mathbf{F}$ and (2) whether the synthetic dataset has preserved these features (and hence, preserved the target's signal) from the raw data. As feature sets, we can try several different sets:

- Naive feature set with simple summary statistics, $\mathbf{F}_{Naive}$
- Histogram feature set that contains the marginal frequency counts of each data attribute, $\mathbf{F}_{Hist}$
- Correlation feature set that encodes pairwise attribute correlations, $\mathbf{F}_{Corr}$

3.3.2 Groundhog Attack

The shadow modeling attack, as detailed in Algorithm 2, has several selectable settings, namely the choice of classification algorithm and the choice of feature set to use. In [19], the authors, after sufficient experimentation with these settings, were able to conduct comparisons of which choices yielded the best performance for the shadow modeling attack. Therefore, we are able to define the "Groundhog attack" simply as a sub-case of the shadow modeling attack for membership inference, where we specifically use a Random Forest classifier with 100 estimators using the Gini impurity splitting criterion. The choice of feature set $\mathbf{F}$ is left open and unconstrained in the Groundhog attack and we are free to explore this aspect further.

CHAPTER 4

Experiments

In this section, we provide an overview of some experimental results we obtain as a result of running a privacy audit using a shadow modeling attack, as described in previous sections, on various algorithms for generating private synthetic data. To perform our experiments, we utilize the open-source Python package `TAPAS` [7]; this framework contains a toolbox of a wide variety of different adversarial attacks, including an implementation of the "Groundhog attack." `TAPAS` also comes equipped with many customizable options for running a privacy audit and computing and evaluating the resulting privacy metrics. We expand upon `TAPAS` and its library of threat model pieces in several ways; our main addition is to introduce the use of other generators not incorporated by the original framework.[1] Namely, `TAPAS` requires a generator that is well-defined within its class structure so that it may repeatedly call said generator during the process of an adversarial audit (i.e., while training an attack), so we integrate our generators of interest (CTGAN, DP-CTGAN, and PATE-GAN) directly into the audit pipeline.

While the main focus of our audits will naturally be on comparing performance across generators to see which may be optimal at preserving privacy, we will indeed also investigate the privacy results across our different datasets and different target records as well. We apply the Groundhog attack in `TAPAS` to multiple different datasets as a comparative study, whereas previous works only look at certain common datasets in isolation. Essentially, we will be evaluating a "matrix" of results in which each dimension/axis corresponds to a different

[1]as of June 12, 2023

aspect of the audit to customize: the generator attacked, the dataset used, and the target record chosen. We begin by describing in detail the different choices we examine for each of these aspects.

4.1 Experiment Setup

4.1.1 Generators

For the purposes of this study, we look at a handful of the most prevalent, widely used generators in research and industry. In particular, we will analyze the resulting privacy of CTGAN, DP-CTGAN, and PATE-GAN. We note that one of the above listed methods, CTGAN, is *not* a differentially private generator, while the other two are as such. Also, we reiterate that we choose three GAN-based methods here; this lies in line with the current state-of-the-art for generative modeling, where much attention has been paid to GANs and their extensions in recent years, at least up until the recent rise of the transformer and its associated models. (Unfortunately, the transformer itself has not yet readily been adapted to our use case of a tabular synthetic data generator, at least with any great success over other classes of models, so we must omit it in our investigations.) In particular, we wish to pay attention especially to PATE-GAN since to the best of our knowledge, this method has not been thoroughly examined for privacy leakage by any existing work – besides some brief mention in [19] – yet is one of the most used synthetic data generators in practice, known both for its high fidelity/utility of synthetic samples and its relative computational efficiency.

For the differentially private methods, we try several different values for the privacy parameter ε, setting three levels $\varepsilon = 0.5, 1, 3$. We do this so that we can analyze how the estimates for ε_{eff}, and therefore the ε_{eff} versus ε relationship, may change for stricter or looser ε constraints on private models. Also, $\varepsilon = 1$ in particular appears to be an accepted baseline of sorts in the existing literature for a strong yet realistic privacy guarantee. We note that for some of the instances, choosing a relatively high ε value causes computational bottlenecks, so we err on the side of smaller ε analyses. For each of the generators, we use the base, open-source implementations and keep the default settings for any tunable hyperparameters, with the exception of decreasing the number of epochs (default 300) for computational efficiency reasons.

4.1.2 Datasets

We consider three datasets in total for our simulation experiments: Census[2], Adult [1], and Texas[3]. We consider these datasets as they each come from a setting in which privacy is extremely important, namely personal demographic data and healthcare data. The Census dataset comes from the Office for National Statistics' 2011 England & Wales 1% Microdata Teaching File and covers various demographic data. The Adult dataset is a classic benchmark for tabular data in machine learning publications and looks at demographic data focused on various predictors that may influence individuals' incomes, which is the variable of interest. The Texas dataset captures hospital stays in the state of Texas and includes measurements such as length of stay and costs.

It is often important when dealing when data synthesis tasks with tabular data to consider other features of the original data, such as the cardinality of categorical attributes, ranges of continuous attributes, quantity of rare/unique real values, and so on. We present detailed information on each dataset at hand below in Table 4.1, and we shall see that such characteristics may play a role in how the nature of the generated data differs on a case-by-case basis in regards to privacy.

The Census dataset contains an anonymized sample of 1% of the responses to the 2011 census conducted by the government of the United Kingdom. It consists of approximately

[2]https://www.ons.gov.uk/census/2011census/2011censusdata/censusmicrodata/microdatateachingfile

[3]https://www.dshs.texas.gov/texas-health-care-information-collection/health-data-researcher-information/texas-inpatient-public-use

570,000 observations of 15 categorical attributes, which are typical census measurements such as education level, family type, region, etc. Each of the categorical attributes has been label encoded so that each possible value of a given attribute is some positive integer, hence the data file has been anonymized since the one-to-one correspondence between the encoded label and the actual feature value is unknown to the public. For computational reasons, we choose to keep only a subset of 10,000 of the original records for our analysis but keep all of the columns intact – running privacy audits with full, large datasets is very computationally expensive!

The Adult dataset is a widely used dataset for classification tasks in machine learning research. It contains anonymized information about individuals from the 1994 U.S. Census database and the goal of models built using the dataset is typically to predict whether a person's income exceeds $50,000 per year based on the various personal attributes. The Adult dataset is often used for training and evaluating machine learning models, as it provides a realistic scenario for predicting income levels based on demographic and socioeconomic attributes. The dataset, much like Census, includes features such as age, education level, marital status, occupation, race, gender, capital gain, capital loss, and more. Lastly, it contains approximately 32,000 records with 18 columns.

The Texas dataset contains data on hospital stays in the state of Texas and includes information such as days stayed and costs of care. This dataset comes in many different forms, but we use the version which contains 1,000 observations (this dataset is sometimes referred to as Texas1000 for this reason) so that we have lower computational cost as well as a different magnitude of dataset size for comparison to the previous datasets. Texas differs in that it contains continuous, numerical features in addition to discrete, categorical ones. To be precise, Texas consists of 18 columns, 6 of which are real-valued and continuous; the 12 discrete columns are a mix of integer-valued and character data.

	# rows	# cols	# discrete
Census	10,000	15	0
Adult	32,000	18	0
Texas	1,000	18	6

Table 4.1: Overview of Dataset Characteristics

4.1.3 Attack Settings

When defining the threat model as a whole, we must designate the value of certain special hyperparameters for the attack; we outline our choices for our experiment runs here. First, when defining the adversary's data knowledge, we designate auxiliary data knowledge, using a 0.5 splitting proportion and dividing the original dataset into two equal parts, one as an auxiliary dataset from which to sample real datasets during the attack and one as the testing set. Each dataset we sample during the attack process, both real and synthetic, consists of 1,000 records, with the exception of Texas, where we only take 100 records due to its smaller size. We generate 100 datasets R_j from the auxiliary data knowledge to train the attacks, meaning we instantiate the linkability/membership inference game, as outlined in Algorithm 1, 100 times. In addition, we generate 100 datasets from the testing set in order to evaluate and test the attack after it has been trained. We keep these settings constant throughout all of the simulations unless stated otherwise.

As our attack function $\mathcal{A}^{\mathcal{L}}(\cdot)$, we use the shadow modeling attack procedure as detailed earlier in this paper; more specifically, we use the Groundhog attack. We use this attack for each of the different generators and datasets for uniformity. As part of the Groundhog attack, we must specify a feature-based set classifier and in turn, we must specify a choice of feature set(s) to use in order to extrapolate features from the high-dimensional data distributions. The feature sets we investigate include the naive set, the histogram set, and the correlation set; we may consider these feature sets individually as in [19], or in conjunction with one another. By default, we use the combination of these three feature sets.

Lastly, to define the threat model for our attack, we must select a target record $\mathbf{r}_t$. In the following experiments, we explore two different classes of these targets: random and outlier. The first is rather self-explanatory, while the second is somewhat open-ended. Finding outlier records in a tabular dataset of mixed columns (i.e., discrete and continuous features) is a high-dimensional problem with many possible choices. For simplicity, we crudely select our outlier records by applying the Isolation Forest algorithm[4], a popular algorithm for outlier detection in tabular datasets, although we could also corroborate these designated outliers with the output of some other outlier detection algorithms. The selection of outliers from a high-dimensional data distribution resulting from a tabular, mixed-type dataset is an open-ended problem with many possible solutions; we present only one such idea here. With that being said, the following analyses predominantly look at randomly chosen targets $\mathbf{r}_t$, unless otherwise noted.

4.2 Results

We are now able to run our Groundhog attack privacy audit on a selected target record from our choice of dataset and choice of synthetic data generator. We now go over some results and take a comparative look at privacy under different such choices. In particular, we will want to compare the performance of different generators since in the end, we would ideally want privacy auditing to help decide which generator is best (in terms of privacy) for a given use case. First, we can apply our attack on each dataset using each of the generators; for these attacks, we select five target records at random. Our choice of generators for each dataset differs slightly due to some computational issues. For example, we must omit PATE-GAN from our attack on Texas and instead choose to replace it with the "Raw" generator; this generator returns a row subsample of the original dataset and can be regarded as the most trivial generator, as it just copies some original records to its "synthetic" dataset. We

[4]We use the implementation in Python's scikit-learn, with no tuning done on the hyperparameters.

can visually compare important statistics such as the classification accuracy and area under the receiver operating characteristic curve (AUC) and the privacy gain (PG), which we previously introduced, for each of these attacks on the three datasets. We plot our results in Figure 4.1, where we evidently see that the attack outcomes differ greatly from one dataset to another. In this plot, the values of the points in the legend simply refer to indices of the dataset which we use to refer to each target $\mathbf{r}_t$.

In the Texas dataset, we see that CTGAN offers virtually no privacy, even when compared to the Raw generator. This signals that the inclusion of any given target $\mathbf{r}_t$ is traceable from the synthetic data to the real data, even more so than if the attacker were simply given some random records directly from the real data. DP-CTGAN, here using $\varepsilon = 1$, does appear to uphold privacy, as the highest AUC we observe for the attack is only just above 0.6. We can also see some instances of an ε_{eff} measurement, particularly for DP-CTGAN. A couple of the points are above the threshold $\varepsilon = 1$, signaling that the privacy promise is not being fully upheld. Not all target record results have an associated estimate of ε_{eff} on these plots; this is due to either a very high ε_{eff} beyond the bounds or numerical issues with floating-point precision, resulting in $\varepsilon_{eff} = \infty$.

The Census and Adult plots, on the other hand, show a rather different pattern. For all of the generators, the attack accuracy is quite low, even for the CTGAN generator, which does not come with any explicit privacy guarantees. This suggests that perhaps rather than the generator choice being the most important component, it is actually the dataset one is using which plays the largest role in privacy evaluations. It appears that the Texas dataset is very susceptible to privacy attacks across the board for different genertors, however the same analysis for Census and Adult shows close to no attack success. This is indicative of a more data-centric nature of privacy preservation than one might have expected; instead of generators having clear advantages or disadvantages, the dataset at hand is the predominant influencer of privacy leakage under adversarial attacks, holding all else constant.

For the Census and Adult datasets we again primarily look at the benchmark level of

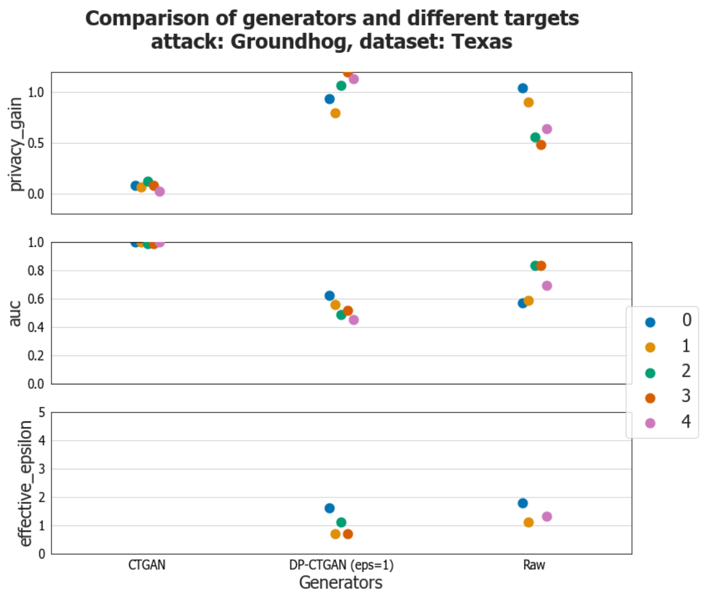

Figure 4.1: Comparing different generators on random targets from Texas dataset.

$\varepsilon = 1$. The resulting privacy gain in these cases is close to 1 with only slight deviations, regardless of the choice of generator. Additionally, the estimated ε_{eff} is not below the theoretical value for all of the instances. In investigating ε_{eff}, we usually want to see values *close to* the theoretical value of ε, not necessarily just lower values. This is because if the ε_{eff} estimates are much lower than ε, this is a potential sign that our privacy analysis is not tight or that the attack used is sub-optimal and requires further selection and tuning. Indeed, in Figures 4.2 and 4.3, some of the ε_{eff} points are much lower than the $\varepsilon = 1$ line, so there may be some further work required to perform a fully tight privacy analysis.

Results such as those we have presented above are straightforward and replicable for different settings and hyperparameter choices using the adversarial auditing tools offered by `TAPAS`; this serves as a template from which we are able to construct other, more detailed and specific analyses. For instance, we have discussed the choice of target record $\mathbf{r}_t$. Since any adversarial attack we run on a generator is by its nature dependent on the target record, this is an influential choice we can make. For each choice of this $\mathbf{r}_t$, we instantiate a given attack

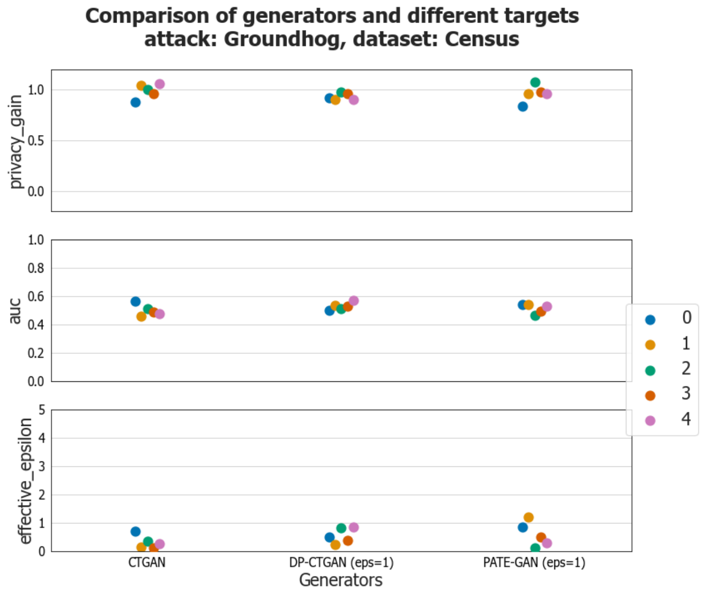

Figure 4.2: Comparing different generators on random targets from Census dataset.

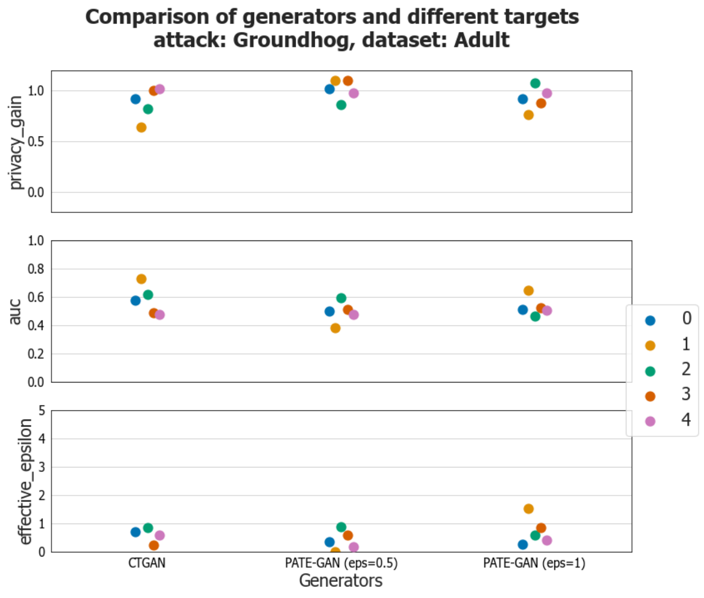

Figure 4.3: Comparing different generators on random targets from Adult dataset.

and receive a set of resulting metrics, such as the aforementioned AUC and privacy gain measurements. Because of this, different choices of $\mathbf{r}_t$ should be expected to yield different levels of resulting attack success; some records in the dataset will be more traceable, even through data synthesis, than others.

For example, it is a natural guess that outlier records would be easier to attack and infer membership of, since we might expect that an outlier has a heavier influence on the resulting synthetic data distribution obtained from a generator than the average random record, and thus be more detectable even after model fitting. We can observe this phenomenon as in Figure 4.4, which displays the attack metrics resulting from attacks on random versus target records from the Texas dataset for the different generators. From these plots, we are able to see that for the differentially private method DP-CTGAN, there is a noticeable difference in privacy gain and attack AUC between attacks on random $\mathbf{r}_t$ and attacks on selected outlier $\mathbf{r}_t$. The disparity between random and outlier targets is largest for the DP-CTGAN model with $\varepsilon = 1$, where the privacy gain is close to 1 for randomly chosen targets and just around 0.7 for outlier targets, which is actually a substantial difference. It is evident that the ε-differential privacy guarantee does not hold evenly across all examples in the dataset and that the differentially private generator sometimes struggles more to "anonymize" these outlier records.

We can again extend this exploration to the other datasets, however, and see that the pattern of outlier targets leaking more privacy than randomly chosen ones does not necessarily always hold. In Figures 4.5 and 4.6, we examine attacks on random versus outlier $\mathbf{r}_t$ for the other two datasets under a suite of generators. Evidently, whether $\mathbf{r}_t$ is an outlier or not does not have the same influence on the final outcome of the privacy audit, compared to our observations for the Texas dataset. This mirrors our observations for the randomly chosen $\mathbf{r}_t$ in Figures 4.1-4.3 – namely, that the Texas dataset audits gave distinctly different results compared to the audits for the Census and Adult datasets. It is apparent that outliers are more apparent to the adversary when attempting to infer membership in R using S in certain

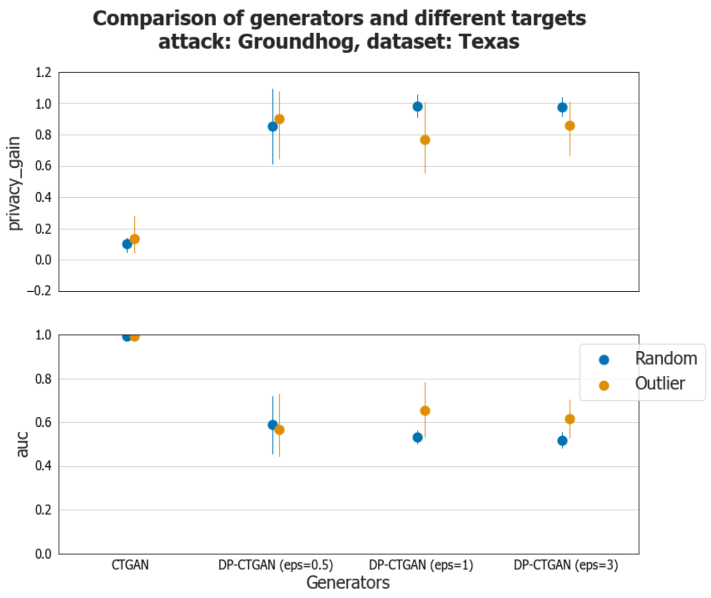

Figure 4.4: Random vs. outlier targets from Texas dataset.

datasets compared to others. This sensitivity to outlier tracing would likely come down to various factors and metadata of a given dataset, such as number of continuous versus discrete columns and quantities such as range of numerical columns and cardinality of categorical ones. Of course, how traceable an outlier is would also have to be assessed while keeping in mind the criteria and methodology used to select said outliers. Overall, we postulate that for datasets such as Texas which have several real-valued, continuous columns, outliers are "farther" away from the center of the high-dimensional data distribution, thereby making them stronger and more extreme outliers than outliers from a data distribution where each dimension corresponds to a categorical column. The cardinality of categorical columns would also likely play a role; if the cardinality of a particular categorical column were to be very large, then the distribution of the column would approach the behavior of a real-valued column.

Lastly, using our experiment setup, we can provide empirical estimates for the value of ε_{eff} for all of the generators we tested and for each dataset. In the estimation of ε_{eff}, we

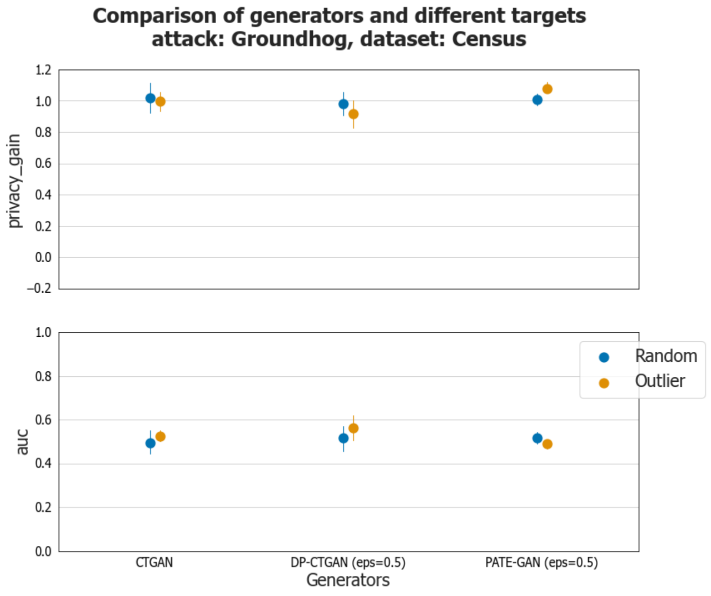

Figure 4.5: Random vs. outlier targets from Census dataset.

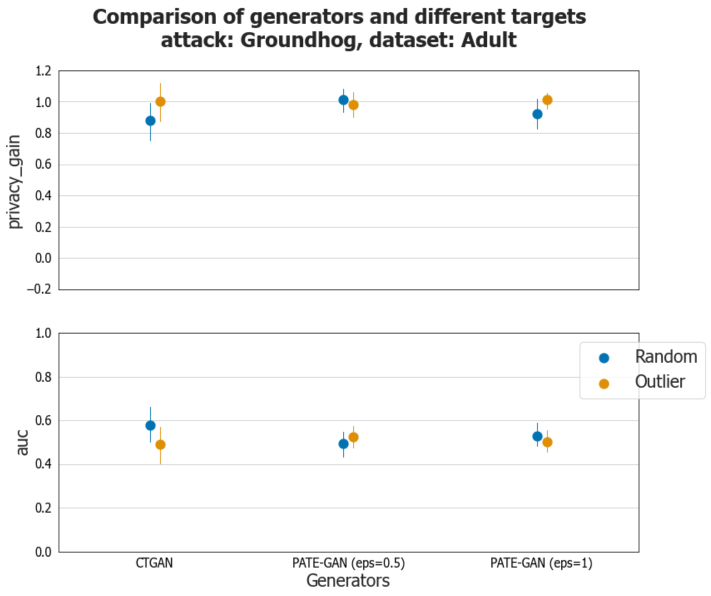

Figure 4.6: Random vs. outlier targets from Adult dataset.

are able to select which conducted attacks to include in the computation of the confidence bounds. For instance, in Table 4.2, we have the results of estimating ε_{eff} for each of the differentially private generators, at each designated ε level of privacy on the different datasets. The estimated values we present in the table simply correspond to the mean/average ε_{eff} we get for all of the attack iterations run on the (dataset, generator) combination at hand. As previously mentioned, we would like to see the ε_{eff} estimates for a given ε-differentially private synthetic data generator be very close to the theoretical value of ε. We note the omission of ε_{eff} results for some cases of (generator, dataset) pairings; in most cases, this is due to computational constraints or, in the case of Texas with the PATE-GAN models, due to the base implementation of this algorithm not being applicable to datasets of smaller size.

	DP $\varepsilon = 0.5$	DP $\varepsilon = 1$	PATE $\varepsilon = 0.5$	PATE $\varepsilon = 1$
Texas	0.374	0.744	-	-
Census	0.605	-	0.756	0.921
Adult	-	-	0.386	0.889

Table 4.2: Effective Epsilon by Dataset and Generator

CHAPTER 5

Conclusion

Privacy auditing is a new but important subfield of synthetic data and serves as a promising, much-needed supplement to theoretical differential privacy. Unlike other areas of secure or private machine learning where results are often empirical observations that appear true without formal justification (e.g., defenses to adversarial examples), the appeal of differential privacy is that it gives provably correct results. The privacy auditing process serves as a useful way to validate the true level of differential privacy of a synthetic data generator and has several advantageous qualities, such as being applicable to checking privacy on a per-observation basis as opposed to the per-sample nature of most or all other privacy metrics (including differential privacy). Even in many quantitative works on privacy, the authors only look at arbitrary cases and surface-level experiments in their investigation of privacy. The main appeal of privacy auditing, if done rigorously, is that it can be used to establish lower bounds on the ε parameter of differential privacy.

We have examined one proposed scheme for privacy auditing of a machine learning model and applied it tp the particular use case of auditing synthetic data generators. While a truly extensive and thorough privacy audit of such complex models would require many runs, a large number of trial datasets and target records, and vast computational resources in order to truly optimize the attacks and approach the desired "worst-case" privacy scenario, we have been able to show in our analysis promising insights into the behavior of some of the most popularly used generators and their privacy guarantees. Naturally, algorithms which offer differential privacy yielded higher privacy gain under adversarial auditing than nonprivate algorithms. Due to various issues, e.g., sometimes implementations of synthetic data generators are flawed, so-called private algorithms did not always cover differential privacy guarantees. We saw this behavior on the individual record level, assessing the privacy gain and membership inference attack accuracy, among other metrics, for each selected target record. We then postulated that outlier records are much more susceptible to attacks and privacy leakage than most randomly chosen records, constituting a disparate vulnerability between different targets under an adversarial attack. While both of these above points is sometimes true, in actuality, much of the divergence in attack performance often came down to characteristics of the source dataset, sometimes even more so than the characteristics of the synthetic data generators themselves.

5.1 Future Work

The most obvious place to start for further studies into this topic is conducting a more robust investigation of privacy attacks to apply; membership inference was chosen due to its prevalence in the privacy literature, however attribute inference, singling out, and reconstruction could also be considered when analyzing the privacy delivered by a particular generator. Even within membership inference and linkability, a truly rigorous audit would apply a panoply of attacks, in addition to the Groundhog attack which we used in this paper, to get closer to the worst-case attack scenario and get a tighter bound on the privacy level ε. The ideal scenario would be to build a library of attacks and have the ability to select and refine attacks based on certain specifications of the problem at hand, such as dataset characteristics and metadata. In fact, analyzing the impact of dataset properties on attack success and overall privacy audit results would also be an avenue for future exploration. In a data-centric view of privacy auditing, it seems that characteristics of the dataset at hand are often an incredibly important aspect of the problem, possibly even more important than the generative model being assessed in the first place. Beyond this, during the actual attack procedure, knowing how to systematically select “vulnerable” outlier records also remains a deep question that is being explored in high-dimensional statistics. Lastly, any practical real-world application of a synthetic data generator would also certainly require careful consideration of that generator’s fidelity or utility on downstream machine learning tasks. While not discussed here, this is a crucial part of analyzing or comparing different generators and much more work is needed to fully understand not just privacy, but also the dynamics between *utility* and privacy, so that we may eventually assess the privacy-utility tradeoff to distinguish between synthetic data generators.

REFERENCES

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[3] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9:211–407, 2014.

[4] Mei Ling Fang, Devendra Singh Dhami, and Kristian Kersting. Dp-ctgan: Differentially private medical data generation using ctgans. In Martin Michalowski, Syed Sibte Raza Abidi, and Samina Abidi, editors, *Artificial Intelligence in Medicine*, pages 178–188, Cham, 2022. Springer International Publishing.

[5] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[6] David Heckerman. A tutorial on learning with bayesian networks, 2022.

[7] Florimond Houssiau, James Jordon, Samuel N. Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum Mole, Camila Rangel-Smith, and Lukasz Szpruch. Tapas: a toolbox for adversarial privacy auditing of synthetic data, 2022.

[8] Bill Howe, Julia Stoyanovich, Haoyue Ping, Bernease Herman, and Matt Gee. Synthetic data for social good, 2017.

[9] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd?, 2020.

[10] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy, 2015.

[11] Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis, 2023.

[12] Bogdan Kulynych, Mohammad Yaghini, Giovanni Cherubin, Michael Veale, and Carmela Troncoso. Disparate vulnerability to membership inference attacks, 2021.

[13] Ryan McKenna, Gerome Miklau, and Daniel Sheldon. Winning the nist contest: A scalable and general approach to differentially private synthetic data, 2021.

[14] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. Aim: An adaptive and iterative mechanism for differentially private synthetic data, 2022.

[15] Casey Meehan, Kamalika Chaudhuri, and Sanjoy Dasgupta. A non-parametric test to detect data-copying in generative models, 2020.

[16] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with pate, 2018.

[17] Haoyue Ping, Julia Stoyanovich, and Bill Howe. Datasynthesizer: Privacy-preserving synthetic datasets. *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017.

[18] R. Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, 2016.

[19] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. Synthetic data – anonymisation groundhog day, 2022.

[20] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms, 2022.

[21] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Neural Information Processing Systems*, 2019.

[22] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2018.

[23] Jinsung Yoon, Lydia Drumright, and Mihaela Schaar. Anonymization through data synthesis using generative adversarial networks (ads-gan). *IEEE Journal of Biomedical and Health Informatics*, PP:1–1, 03 2020.

[24] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2019.

[25] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Trans. Database Syst.*, 42(4), oct 2017.