SUMMARY

Data sharing has become an important aspect in the development of new predictive models. This includes, for example, having access to real healthcare data for the development of disease detection models. However, privacy regulations limit access to such data as to prevent private data from being exposed to the public and most importantly from individuals with malicious intent. To allow for data sharing with privacy guarantees, there are commonly three different ways; Policy, cryptography and anonymisation. Policy means limiting access to the data so only certain individuals can access the data or just parts of it. Cryptography employs encryption algorithms to make the data unintelligible and unusable for anyone without a proper decryption key. Anonymisation techniques focus on altering or removing identifiable information from datasets while retaining their utility for analysis and research purposes.

Anonymisation techniques have shown a lot of promise, especially in the context of generative models such as GANs (Generative Adversarial Networks) and VAEs (Variational Autoencoders). These models employ some type of privacy mechanism in order to obfuscate the data. One of the most used mechanism is differential privacy, which introduces a specific amount of noise into the data determined by the parameter ϵ. With this noise, certain guarantees are made for the privacy of the dataset. One of the benefits of differential privacy over prior privacy mechanisms is that you no longer need attack modeling. This means that no assumptions need to be made of what the attacker knows, rather, a guarantee is made for how much information the attacker can gain from the anonymised data. However, for e.g. verification purposes it may still be helpful to do attack modeling.

Despite the promise of anonymisation techniques, the literature lacks a comprehensive evaluation methodology. Current literature predominantly focus on utility metrics, leaving the privacy aspects under-explored. Additionally, they often have a limited scope, in terms of only evaluating a subset of privacy attack types. To address this issue, our paper aims to evaluate the state of the art privacy metrics, encompassing relevant privacy attacks, in order to find the most essential privacy metrics for measuring privacy of anonymised datasets.

We conduct such an evaluation with two sets of experiments. The first set of experiments aims to evaluate whether the metrics work as expected. This entails firstly establishing upper- and lower-bound baselines and checking if all other results fall in-between these, with which we confirm whether the metrics' extreme-case scenarios are as expected. After this, we check the correlation between the anonymisation parameter ϵ and the results of the metrics, for which we assume that changing ϵ gives appropriately anonymised datasets, in order to further test the metrics. The second set of experiments aims to select a subset of the metrics that are sufficient for evaluating the privacy of an anonymised dataset. For this, the correlation is calculated between each pair of metrics to test if any can be excluded based on being highly correlated with another, and clustering of the metrics is performed to identify groups of metrics that are similar. The data used for this correlation and clustering is the full results of the prior set of experiments, which means the data points for each metric is its result for each generated anonymised dataset, meaning we are identifying similarity based on the outputs of the metrics.

As a result of the experiments we went from 21 to 7 metrics, which we deem to be sufficient for evaluating the privacy of anonymised data. The results showed that these 7 are able to capture the anonymisation level of the anonymisation techniques to an adequate degree, including when changing ϵ for DP techniques. Computing the correlation between the metrics did not help much in determining similarity between the metrics. However, the clustering results were more helpful which further helped narrow down the metrics. Both the correlation and clustering results however do not seem to be according to our categorisation of the metrics into attack categories.

The limitations of these results primarily come down to how well-tuned the anonymisation techniques and metrics are, where more time could have been spent tuning them. The metric descriptions could also be improved, where fewer initial metrics could have been sufficient, which is also found based on other reasons. Further work also includes testing on more anonymisation techniques and datasets, and exploring e.g. Model Extraction.

With the two sets of experiments, we were able to exclude the metrics that either performed inadequately or which use case was covered by another metric. This gave a list of 7 metrics deemed adequate for testing anonymised data, covering all three attack categories.

A Comparison of Privacy Metrics for Synthetic Data Generation

ASTRID MELODI HANSEN, Department of Computer Science, Aalborg University, Denmark
FREDERIK STÆR, Department of Computer Science, Aalborg University, Denmark

ABSTRACT

Data sharing has become a major factor in the development of robust new machine learning models, especially, in the health sector for e.g. disease prediction. However, sharing such data presents a privacy risks for individuals present in the data. Therefore, privacy laws have been introduced to protect such individuals, those protections being GDPR (General Data Protection Regulation) and HIPAA (Health Insurance Portability and Accountability Act). In the context of data sharing, however, this makes it difficult to share data between institutions. To overcome this issue anonymisation techniques has been suggested to anonymise the data.

Anonymisation techniques are essential in safeguarding sensitive data while still allowing its utilisation for research, analysis, and other purposes. These techniques aim to remove or obscure personally identifiable information from datasets, thus reducing the risk leaking sensitive information while preserving the data's utility. Several anonymisation methods exist, each with its strengths, limitations, and suitability for different data types and use cases.

Evaluating anonymisation techniques usually revolves around testing the utility and privacy of the anonymised data. However, in the current literature not much attention has been paid to testing privacy with some papers only testing the utility of the anonymised dataset and others only testing a limited number of privacy attacks.

Therefore, in this paper, we evaluate the state of the art privacy metrics, covering different privacy attacks, in order to establish which privacy metrics are necessary to thoroughly test anonymised datasets.

We perform two different sets of experiments. The first is aimed at testing whether the privacy metrics work as expected, which is measured as whether the privacy level of a given anonymisation technique correlates with the score of a given privacy metric. The second investigates whether different privacy metrics capture different aspects in terms of the anonymisation. This is measured by calculating the correlation between the scores of the individual privacy metrics and by performing clustering on these scores. The experiments are conducted on two different tabular datasets; MedOnc (8,630 rows) and Texas (25,000 rows).

Through a metric selection process, the results showed that 7 out of 21 metrics (1) worked as expected to some degree and (2) are able to capture different aspects of anonymisation. These metrics are therefore deemed sufficient for evaluating the privacy of anonymised data in the context of tabular data.

Authors' addresses: Astrid Melodi Hansen, Department of Computer Science, Aalborg University, Aalborg Ø, Denmark, amha17@student.aau.dk; Frederik Stær, Department of Computer Science, Aalborg University, Aalborg Ø, Denmark, fstar19@student.aau.dk.

1 INTRODUCTION

Data sharing has increasingly become an important part in the search of new scientific discoveries, such as in disease detection research. Privacy regulations such as GDPR (General Data Protection Regulation) and HIPAA (Health Insurance Portability and Accountability Act) limit this to protect private information included in the data. Therefore, to be able to share data with others this needs to be done in a secure way. Al-Aqeeli et al. [1] define three different approaches to ensure this: policy, cryptography and anonymisation. Policy relates to limiting access to the data in the form of security roles. Cryptography revolves around encrypting the data, making it unusable for attackers. Anonymisation attempts to either modify the sensitive data utilising a specific technique, such as row-level sanitisation, or generate new anonymised data using generative models such as GANs (Generative Adversarial Networks) [2]. Row-level sanitisation techniques however suffer from certain privacy attacks especially when a given dataset has high dimensionality [3]. Generative models (synthesisers) have especially shown great promise in this regard, incorporating privacy mechanisms such as differential privacy.

When introducing new synthesisers, there are usually two different aspects that needs to be evaluated: utility and privacy. Utility refers to the data being usable e.g. it has a similar data distribution as the original data. Privacy relates to the data being protected in terms of an attacker not being able to infer sensitive information using privacy attacks [4].

For simplicity only tabular data is considered in this paper, containing either categorical or numerical values. Here, certain EHR (Electronic Health Record) data is also included in tabular data.

In the literature on synthesisers, there is usually more emphasis on utility, sometimes not even considering privacy [5, 6]. This could be due to them believing if e.g. their synthesiser employ differential privacy, then the anonymised data must be private. Furthermore, there exists a number of different privacy attacks an attacker can perform, but some of the literature only considers some of these. Therefore, no comprehensive evaluation approach is used throughout the literature, making it hard to compare results [7].

Some literature [8–10] has tried to make a comprehensive evaluation framework where a number of different privacy metrics are included in order to test the level of privacy of a given anonymised dataset. However, they do not make an explicit distinction between a given privacy metric and the privacy attack it measures, and thereby potentially do not cover all privacy attacks. Additionally, they do not specify if a subset of privacy metrics is adequate in order to evaluate the privacy of the anonymised dataset. For example, instead of multiple privacy metrics, simply using one might be sufficient. Based on this, we have the following problem statement:

In the context of tabular data:

(1) *how well do existing privacy metrics cover relevant privacy attacks?*
(2) *do existing privacy metrics capture the privacy preservation of anonymisation techniques?*

This problem statement defines two different questions. For the first part we test this by measuring the correlation between the individual privacy metric scores, and here it is expected that metrics in the same attack category are more correlated. For the second part we experiment with a number of different synthesisers having different levels of privacy guarantees. Here, we expect the privacy metrics to reflect this i.e. synthesisers with high privacy guarantees getting good privacy scores (being more protected from attacks) and vice versa.

To answer the problem statement we perform an experimental comparative analysis of 21 privacy metrics. The privacy metrics included in this paper were initially sourced through a systematic review, conducted in previous work [7]. After this, some metrics were removed due to not functioning well enough, and new metrics were found in further examination during the making of this paper.

1.1 Bibliographical remarks

In this paper, we build upon the findings established in our previous work Hansen et al. [7]. Overall, the previous work provides a foundation for the research presented in this paper. Here, we change, extend and entirely rework the different sections. Our previous work investigated privacy metrics in order to determine the coverage of these in relation to privacy attacks, as well as which privacy metrics should be used in the context of evaluating the privacy of anonymised data. However, the results of the experiments conducted were partially inconclusive.

The abstract has been slightly reworked, but overall content remains the same.

In Section 1, we have reformulated the entire material, including a change to the problem formulation, where the focus of this paper is not EHR data specifically but rather tabular data in general, which includes some EHR data.

In Section 2, we reworked the material and also added further related work, those being Hyrup et al. [11] and Lautrup et al [12]. Additionally, a table comparing the individual studies has been included.

In Section 3, the definition of a dataset has been changed to clarify how attributes work and make it easier to use in metric definitions, and the problems studied have been simplified and defined formally.

Section 4 has been expanded with added content describing differential privacy, anonymisation techniques and privacy metric types. Additionally, the privacy attack descriptions are changed in terms of introducing mathematical definitions and notation. The running example throughout this section has also been slightly updated in terms of being more clarifying of how the individual attacks work. For example a third individual for tracing attacks is introduced and values (e.g. naming) of the individuals have been changed.

Section 5 has been altered in terms of excluding some old metrics as well as adding more new metrics. For the exclusions, this was based on them seemingly not working, and additions have been made partly to make up for this, and partly to increase the scope on this part. Pseudocode has been defined for the metrics, and some metadata for the metrics has been included, such as the type of metric.

In Section 6, we extend the previous research by mainly restructuring and extending the experiments. Here, we expand the number of datasets, synthesisers and privacy metrics used in the experiments. Furthermore, what was Experiment 3 (now the second set of experiments) is expanded with clustering.

Sections 7, 8 and 9 are entirely rewritten, with entirely new data.

Section A is entirely new with an overview of how to preprocess the datasets used in the experiments, as well as the raw results presented again but with colours according to individual values, to aid in discussing clustering results.

2 RELATED WORK

Multiple studies investigate the evaluation of synthesisers, where some provide benchmark frameworks for this purpose. In these, privacy metrics are also included.

Hyrup et al. [11] investigate privacy metrics in the context of determining which privacy metrics are appropriate for anonymised data evaluation. They aim to find the best universal privacy metric, where a total of five privacy metrics are tested. According to Hyrup et al., the universality of a metric is defined by how well it lives up to the four CAIR (Comparability, Applicability, Interpretability, Representativeness) principles. For example, high Interpretability allows non-technical stakeholders to understand to which extent an anonymised dataset is private. Additionally, Hyrup et al. focus on a limited number of privacy metrics (specifically five) and do not consider the different privacy attacks and whether the privacy metrics they investigate actually capture these.

Yan et al. [8] evaluate five different synthesisers using a set of metrics grouped into utility or privacy, testing whether these metrics are adequate to measure utility and privacy. However, they do not indicate which privacy attacks they evaluate. They focus on EHR data whereas we have a different scope of evaluating tabular data, which includes tabular EHR data.

Similar to Yan et al. is Lautrup et al. [12], which is another benchmarking framework aimed at testing the utility and privacy of anonymised datasets.

Qian et al. [9] provide an open-source benchmarking framework, which covers a wide variety of tabular datasets, synthesisers and metrics. This is intended for full evaluation of synthesisers, defined by metrics in seven different aspects: sanity, statistical, performance, detection, privacy, attacks and weighted metrics. The contribution of Qian et al. is a framework allowing for extensive benchmarking of synthetic data.

Similarly to Qian et al., Díaz and García [10] provide a benchmarking framework, just with a focus on anonymity. This framework includes metrics such as k-anonymity, l-diversity, δ-disclosure privacy among others. In our paper, we specifically investigate synthesisers (e.g. DP-GAN) whereas Díaz and García utilise the data anonymisation tool ARX [13] to simply modify the sensitive sensitive data.

Our paper differs from these primarily in the fact that we evaluate the metrics in relation to synthesisers and each other. More differences are shown in Table 1.

point	**Hansen & Stær**	**Hyrup et al.**	**Yan et al.**	**Lautrup et al.**	**Qian et al.**	**Díaz & García**
Tabular data	X	?	X	X	X	X
EHR data	/	?	X	X	X	X
Assesses synthesisers	/		X	X	X	X
Evaluates utility			X	X	X	
Evaluates privacy	X	X	X	X	X	X
Assesses privacy metrics	X	X				
Considers privacy attacks	X				/	

Table 1. Which points are fulfilled, for each paper. Fulfilment is marked with "X", "/" indicates partial fulfilment and "?" refers to unknown fulfilment.

3 PROBLEM DEFINITION

The datasets in this paper are inspired by tables from the relational model [14]. To define a dataset, we first specify a set of attributes $A = \{a_1, ..., a_m\}$, similar to a header. Each attribute is a set of allowable values (a data type), specifically either $\mathbb{R}$ (numerical), $\sum^*$ for some alphabet $\sum$ (categorical) or $\{0, 1\}$ (binary). A dataset D is then defined as a subset of all combinations of attribute values: $D \subseteq a_1 \times ... \times a_m$, similar to the body of a relational database. Each element of D is a tuple, also commonly called a row or record. To access the values of a tuple $t \in D$, we index t on the set of attributes we want to access, e.g. $t[A]$ to get all the values of tuple t; alternatively, $t[a]$ for an attribute a gives the value for that attribute.

Let $\mathcal{D}$ be the set of all datasets. Similarly to the definition proposed by Desfontaines' [15], we define an anonymisation technique $S : \mathcal{D} \times \Theta \rightarrow \mathcal{D}$ as a function that maps a sensitive dataset Y and a set of parameters θ, to an anonymised dataset Z, i.e.: $Z = S(Y, \theta)$. Z is the dataset that can be released, and that is available to the data analysts (including potential attackers).

A privacy metric $P : \mathcal{D} \times \mathcal{D} \rightarrow [0, 1]$ maps a sensitive dataset and an anonymised version to a score in the range $[0, 1]$, with a score of 0 indicating complete privacy and 1 indicating no privacy. For example, let D be a sensitive dataset, and *Real* the identity function, in which case $P(D, Real(D, \theta)) \simeq 1$. We indicate with $\mathcal{P}$ the set of available metrics.

The model of attack used in this paper is the black box model, which means that the attacker only has access to $S(D, \theta)$ and some existing knowledge [16]. Existing knowledge is some known attributes about a given person, such as name, age and profession. This attack model reflects the data-sharing problem as mentioned in Section 1, as anonymisation is the focus, where a new dataset is generated and only that dataset is released.

The problems studied in this paper are:

(I) Do the metrics work as expected, i.e. Is it true that $P(D, S_1(D, \theta_1)) \leq P(D, S_2(D, \theta_2))$ when (S_1, θ_1) is more anonymous than (S_2, θ_2)?

(II) Do different metrics capture different aspects of the anonymisation? That is, for any metrics P_1 and P_2, is it true that $P_1(D, S(D, \theta)) \nsim P_2(D, S(D, \theta))$ for some S, θ across all D?

4 BACKGROUND

4.1 Differential Privacy

DP (Differential privacy) is a mathematical notion to ensure the anonymity of individuals in a dataset [17]. Here, an algorithm is said to be differentially private if Equation 1 holds.

$$Pr[M(D_1) \in S] \leq exp(\epsilon) \cdot Pr[M(D_2) \in S] + \delta \tag{1}$$

In Equation 1, $Pr[M(D_1) \in S]$ describes the probability that a randomised algorithm M applied to dataset D_1 produces an output in set S, while $Pr[M(D_2) \in S]$ denotes the same for a neighbouring dataset D_2. A neighbouring dataset is a dataset that differs by only one tuple. This inequality asserts that the impact of adding or removing an individual tuple on the algorithm's output distribution is bounded by a multiplicative factor exp(ϵ) and additive value δ. This in turn ensures a certain level of privacy. ϵ is the parameter for the privacy budget, while δ defines the risk of going over this budget.

An ϵ of 0 indicates complete protection i.e. the data is completely random. However, having the data be completely random makes the data unusable from a utility point of view. Therefore, a privacy-utility balance needs to be established

in terms of how private a given dataset needs to be for it to still be usable for a given use case. Commonly a value between 0.1 and 10 is used.

δ is an additional parameter specifically for handling edge cases where a different privacy guarantee is needed, allowing for a small probability of failure in ensuring privacy. It can be likened to permitting a degree of outliers, where a slight relaxation of the privacy constraint is acceptable in certain scenarios. Moreover, in applications involving high-dimensional data, mechanisms with δ such as Gaussian mechanisms tend to perform better, offering improved privacy guarantees while maintaining utility, particularly in settings where traditional differential privacy mechanisms may struggle to preserve privacy effectively.

4.2 Anonymisation Techniques

Anonymisation techniques take as input a sensitive dataset and outputs an anonymised dataset. There are two types of anonymisation techniques: row-level sanitisation and synthesisers.

Row-level sanitisation revolves around editing the sensitive data using some underlying method. For example, k-anonymity is an approach in which the objective is to have a given row (individual) be identical to at least $k - 1$ other rows on the quasi-identifying attributes [18], therefore providing a certain level of privacy. This is commonly implemented by either generalisation or suppression. However, row-level sanitisation such as k-anonymity has been proven to not perform well against certain types of privacy attacks, especially when an attacker has external background information of specific individuals in a given dataset [3]. Unlike row-level sanitisation, generative techniques show a lot of promise in this regard.

Synthesisers encompass approaches to generate new data, not simply modifying it. This is implemented using generative models such as VAEs (Variational Autoencoders) and GANs (Generative Adversarial Networks). Such models are trained to capture the underlying data distribution of a given dataset in order to generate a new dataset. However, to attain an anonymised dataset using generative techniques, some underlying anonymisation mechanism needs to be employed. DP is a commonly used mechanism of these, where it is integrated into the training of a given model. For example with the model PATE-GAN [19], a certain amount of noise is introduced to gradients when backpropagation is conducted, in order to achieve a specific level of anonymisation.

4.3 Privacy attacks

Dwork et al. [4], as well as Rigaki and Garcia [20], categorise privacy attacks in five different groups: reconstruction attacks, re-identification/de-anonymisation attacks, tracing attacks, correlation detection and model extraction/stealing. We only consider the first three, as correlation detection is not considered a privacy concern [4], and model extraction/stealing does not follow the black-box model assumption we introduced in Section 3. These last two were described in the discussion by Hansen et al. [7].

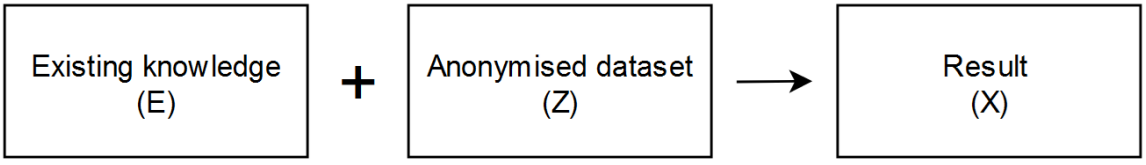

Fig. 1. Example of the general structure of a privacy attack, where existing knowledge (E) is combined with an anonymised dataset (Z) to derive the result (X), which in a successful attack is a subset of the sensitive dataset.

To illustrate the different privacy attacks, which are based on the generic model shown in Figure 1, we consider a privacy attack as having two inputs: An anonymised dataset (Z) and some existing knowledge (E). Using these two inputs into some function f, the output X can be computed which is the result of the privacy attack. Formally, a privacy attack can be defined as a mapping function $f : E \times Z \rightarrow X$

In this context, we consider the individual named *John*, and assume we have some existing knowledge (E) of *John* as well as an anonymised dataset (Z). The anonymised dataset is generated using a synthesiser applied to the sensitive dataset (Y), which contains *John*. Imagining ourselves as attackers, we attempt to perform privacy attacks on *John* to obtain some information about *John* we had no prior knowledge of. Performing a privacy attack yields a result (X), potentially revealing new private information about *John*.

4.3.1 Reconstruction attack. Originally, the reconstruction attack was developed by Dinur and Nissim [21] as a means to show that statistical releases were not sufficient to protect privacy. Later, Dwork et al. [4] adapted the attack definitions such that a reconstruction attack attempts to determine the value of a specific attribute for a given tuple in a dataset.

Our definition for a reconstruction attack is based on how most privacy metrics in Section 5 handle the attack. The difference from Dwork et al. [4] is that there may be several attributes we are reconstructing. This means an attacker has access to an anonymised dataset and some existing knowledge. The goal of the attacker is to infer the missing attribute(s) of the existing knowledge, using both the existing knowledge and the anonymised dataset.

We define a reconstruction attack formally as:

Definition 4.1. Given an external dataset $E = \{e_1, ..., e_{n_1}\}$ with attributes $A = \{a_1, ..., a_m\}$, and an anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$ with attributes $A \cup \{a_{m+1}\}$, a reconstruction attack aims to infer the values of a_{m+1} for the records in E. Specifically, a reconstruction attack is a function f that takes as arguments E and Z and returns a new dataset $X = \{x_1, ..., x_{n_1}\}$ where $x_i[A] = e_i[A]$ for each tuple in E, and $x_i[a_{m+1}]$ is the predicted value associated to the record e_i for the attribute a_{m+1}.

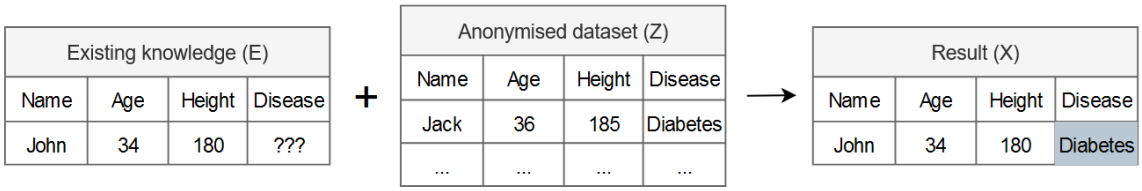

Fig. 2. Example of a reconstruction attack, where we have the sensitive individual *John* and want to find out what the value of the missing attribute *Disease* is. Additionally, we have an anonymised dataset containing the individual *Jack*, among others. Here, it can be inferred that since *John* has a *Height* and *Age* similar to the individual *Jack*, they probably have the same *Disease* attribute, *Diabetes*

An example of a reconstruction attack can be seen in Figure 2. For this example, the objective of the attack is to infer the missing attribute *Disease*. Implementations of such an attack tend to use an equivalence class to calculate the risk for each tuple in the sensitive dataset, such as a KNN algorithm in Yan et al. [8]. This is calculated for each sensitive tuple, and the results are combined in some way, such as by averaging to infer the missing attribute(s).

4.3.2 Tracing attack. Tracing attacks attempt to infer whether an individual is part of the sensitive dataset used to generate the anonymised dataset. This can be useful for attackers, as simply knowing whether an individual is included in a dataset or not can be sensitive information, which can lead to further attacks being conducted with this in mind. Again, an attacker only has access to some existing knowledge of an individual and an anonymised dataset.

We define a tracing attack formally as:

Definition 4.2. Given an external (E) and anonymised dataset (Z) $E = \{e_1, ..., e_{n_1}\}$ and $Z = \{z_1, ..., z_{n_2}\}$, both with a subset of the attributes $A = \{a_1, ..., a_m\}$, we want to infer whether $e_k \in Z$, which is whether a given individual tuple is in the anonymised dataset, or more specifically if the individual was part of the sensitive dataset used to generate the anonymised dataset. A function f to infer whether an individual is part of Z can be defined as $f(e_k, Z) = Pr(e_k \in Z)$, with the output being a probability (Pr) measure.

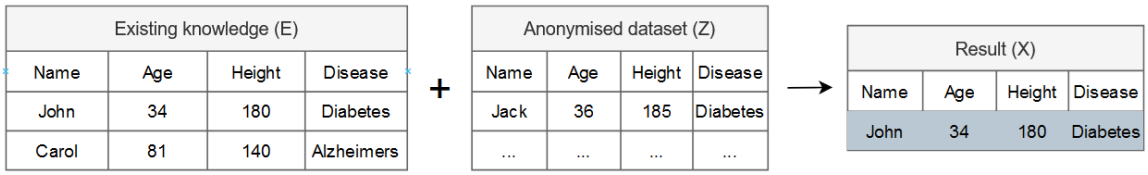

Fig. 3. Example of a tracing attack. In this case we have external knowledge consisting of two tuples i.e. two sensitive individuals *John* and *Carol*, and an anonymised dataset. We want to find out whether *John* and *Carol* are in the anonymised dataset, or rather, in the sensitive dataset used to generate the anonymised dataset. Looking at the *height*, *age* and *disease* it can be surmised that *John* has very close attribute values to those of the anonymised individual *Jack*. Therefore, we deem that *John* is in the dataset. No close match was found for *Carol*, and it is therefore deemed that *Carol* is not in the dataset, completing the attack

Figure 3 depicts a tracing attack. Here we have a sensitive tuple and an anonymised dataset and want to identify whether the individual is in the anonymised dataset. There are a number of different approaches to infer this. Yan et al. [8] perform such an attack by calculating the Euclidean distance between the sensitive tuple and the anonymised tuples. They define a distance threshold, specifying that if the minimal distance is lower than the threshold, we can infer that the sensitive tuple is actually in the anonymised dataset. Note that multiple anonymised tuples could be inside this threshold, but regardless, the assumption remains that the sensitive tuple is in the anonymised dataset.

4.3.3 Re-identification/de-anonymisation attack. Re-identification/de-anonymisation attempts to classify which tuple in an anonymised dataset belongs to a specific sensitive individual. This is done by having access to existing knowledge of the individual, and then perchance linking them to a specific tuple in the anonymised dataset, with the purpose of gaining more information about them. An important note is that the external knowledge is incomplete/partial relative to the anonymised dataset, meaning they do not have the exact same attributes.

Different from tracing attacks is that re-identification/de-anonymisation attacks assume the individual is in the dataset, and attempts find which row(s) specifically belongs to them. We define a re-identification/de-anonymisation attacks formally as:

Definition 4.3. Given an external (E) and anonymised dataset (Z) $E = \{e_1, ..., e_{n_1}\}$ and $Z = \{z_1, ..., z_{n_2}\}$, both with attributes $A = \{a_1, ..., a_m\}$ we want to find for any k at which j it holds that $e_k = z_j$. For this, a function f can be defined to infer which $e_k = z_j$, where the output is a probability (Pr) measure, $f(e_k, Z) = Pr(e_k = z_j)$ for each j.

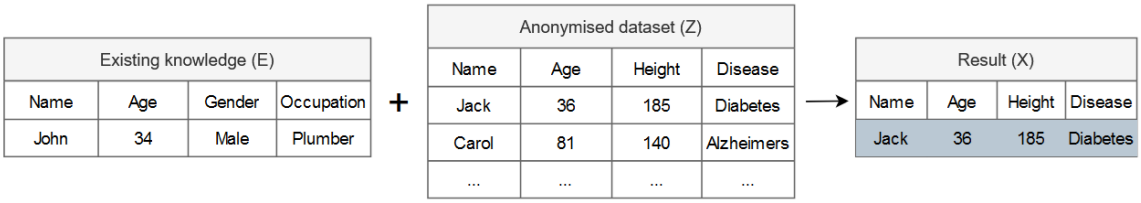

Fig. 4. Example of a Re-identification/de-anonymisation attack, where we have existing knowledge of the individual *John* and an anonymised dataset consisting of multiple anonymised individuals. Given that the only comparative attribute between the external knowledge and the anonymised dataset is the age, we can surmise that *John* potentially is *Jack* in the anonymised dataset.

Figure 4 depicts a re-identification/de-anonymisation attack. There exist a number of different approaches to perform such an attack. Yoon et al. [22] describe an approach in which a classifier is trained on the external data. The trained classifier can then be used on the anonymised dataset predicting the probability of a given individual in the anonymised dataset corresponding to a sensitive individual in the external knowledge dataset.

4.4 Privacy metric types

Separate from privacy attacks, privacy metrics also fall into different types, based on the approach used to measure privacy. The approaches identified through our investigation of the current state of the art privacy metrics are; Distance, Probabilistic and Statistical measurements.

Distance aims to project the data into a space from which a distance between the individual data points can be computed. In many metrics this approach is used using a KNN algorithm [23] to e.g. identify the most similar anonymised tuple of a given sensitive tuple.

Probabilistic metrics aim to evaluate privacy based on the likelihood of an attacker correctly guessing or inferring sensitive information about individuals. These metrics often involve using machine learning classifiers that output probabilistic predictions. For example an MLP (MultiLayer Perceptron) can be employed to measure the re-identification risk by training on sensitive and anonymised datasets and then using the classifier's probabilistic outputs to determine the likelihood of correctly identifying anonymised data points.

Statistical privacy metrics rely on statistical properties and summaries of the data to assess privacy. These metrics often compare statistical aggregates, such as means, variances, and distributions between the sensitive and anonymised datasets to determine how much information has been preserved or leaked.

For all of these types, the distance/probability/statistic may be used in e.g. a ratio or passed through further processing, though this does not change their type.

5 METRICS

In order to conduct an investigation of the state of the art privacy metrics, we previously performed a systematic review of privacy metrics, where we extracted 14 metrics (Hansen et al. [7]). We used these metrics to start with, whereafter some were removed due to not working well enough, and more were added during the making of this paper. The metrics have all been implemented, where a lower score means more anonymised and vice versa. Tables 2, 3 and 4, provide an overview of the metrics for a given privacy attack. Here, computation time is included for each privacy metric as to provide an understanding of the efficiency and scalability of the given metric. The scalability can be partly deduced based on the differences between the datasets, which are described in Section 6.1. Note however that the hardware also varies by dataset, which is described in Section 6.2.

Throughout these metrics, we often use KNN [23]. The way we use this in the algorithms is that KNN$(t, T, A, k = n)$ returns the list of nearest neighbours of t: $[t_1, ..., t_n]$ from the dataset T, based on the attributes A.

5.1 Reconstruction metric algorithms

Reconstruction metrics attempt to reconstruct a given attribute in the external knowledge (subset of the sensitive dataset) dataset based on the synthetic dataset. A commonality with the sourced reconstruction metrics is that key attributes (A_K) and sensitive attributes (A_S) are needed as input. Formally, they can be described as $A_K, A_S \subset \{a_1, ..., a_m\}$ and $A_K \cap A_S = \emptyset$. An overview of these metrics is given in Table 2, after which the individual metrics are further described.

Metric	Type of metric	Applicable attribute types	Additional inputs required	Computation time
Attribute Inference Risk [8]	Statistical	Any	Attributes A, Key attributes A_K, Sensitive attributes A_S	Texas: $1297s \pm 841$ MedOnc: $1226s \pm 14.5$
Categorical ZeroCAP* [1] [24]	Statistical	Any; Works best with categorical	Key attributes A_K, Sensitive attributes A_S	Texas: $185s \pm 6$ MedOnc: $24s \pm 0.27$
Categorical GeneralizedCAP* [1] [24]	Statistical	Any; Works best with categorical	Key attributes A_K, Sensitive attributes A_S	Texas: $1679s \pm 28$ MedOnc: $111s \pm 0.23$

Table 2. Reconstruction metrics used to conduct experiments. These are based upon the findings from the review performed in our previous work [7], as well as metrics found in further examination during the making of this paper marked with *

Algorithm 1 computes attribute inference by first calculating the total entropy of the sensitive dataset Y, in line 2, and then computing attribute weights based on the entropy contribution of each attribute, in line 3. Next, from line 7 to 14, for each tuple y in Y, and for each attribute a in A (the set of all attributes), the equivalence class T_{eq} of y in the anonymised dataset Z is determined. Depending on the type of attribute (a being numerical, categorical, or binary),

[1] The output of this metric has been reversed ($score = 1 - originalscore$)
[2] The output of this metric has been normalised to the range $[0, 1]$
[3] Assumes each anonymised tuple is generated from a single sensitive tuple

the algorithm computes T_{eq} differently. Here, common for all attribute types, is the use of a KNN algorithm [23] to find the nearest neighbours/tuples. For numerical attributes, T_{eq} is determined by whether the attribute value is in the computed thresholds. For categorical and binary, it simply does an equivalence check.

After obtaining the equivalence class, the algorithm calculates the number of true positives (TP), false positives (FP), and false negatives (FN) for each attribute in line 15, 16 and 17, respectively. Then, it updates the attribute inference score $AIScore$ using the F1 score weighted by the attribute weights in line 18. Finally, it returns the final attribute inference score in line 21.

Algorithm 1 AttributeInference

Input: Set of attributes $A = \{a_1, ..., a_m\}$, Key attributes A_K, Sensitive attributes A_S, Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$

Output: A risk score in the range $[0, 1]$

1: $AIScore \leftarrow 0$
2: $totalEntropy \leftarrow$ CalculateTotalEntropy(Y)
3: $attributeWeights \leftarrow$ CalculateAttributeWeights$(totalEntropy, Y)$
4: **for** y in Y **do**
5: $\quad T_{eq} \leftarrow \{\}$
6: $\quad$ **for** i from 1 to $|A|$ **do**
7: $\quad\quad$ **if** a_i is numerical **then**
8: $\quad\quad\quad lowerThreshold \leftarrow 0.9, upperThreshold \leftarrow 1.1$
9: $\quad\quad\quad NN \leftarrow \text{KNN}(y, Z, A_K, k = 1)$ ▷ Get nearest neighbours using k-nearest neighbours algorithm
10: $\quad\quad\quad T_{eq} \leftarrow$ From NN tuples, get the tuple within the thresholds for $y[a_i]$
11: $\quad\quad$ **else if** a_i is categorical or binary **then**
12: $\quad\quad\quad NN \leftarrow \text{KNN}(y, Z, A_K, k = 1)$
13: $\quad\quad\quad T_{eq} \leftarrow \{z | z \in NN \wedge y[A_K] = z[A_K]\}$
14: $\quad\quad$ **end if**
15: $\quad\quad TP \leftarrow |\{z | z \in T_{eq} \wedge y[A_S] = z[A_S]\}|$ ▷ True positive
16: $\quad\quad FP \leftarrow |\{z | z \in T_{eq} \wedge y[A_S] \neq z[A_S]\}|$ ▷ False positive
17: $\quad\quad FN \leftarrow FN + |T_{eq}| - TP$ ▷ False negative
18: $\quad\quad AIScore \leftarrow AIScore + (\text{F1Score}(TP, FP, FN) \times attributeWeights[i])$
19: $\quad$ **end for**
20: **end for**
21: **return** $AIScore$ ▷ Get final attribute inference score
22:
23: **function** CalculateTotalEntropy(Y)
24: $\quad N \leftarrow |Y|$
25: $\quad totalEntropy \leftarrow 0$
26: $\quad$ **for** y_i in Y **do**
27: $\quad\quad p_i \leftarrow \sum_{j=1}^{|Y|} 1(y_i = y_j)$ ▷ Number of instances of y_i in Y
28: $\quad\quad totalEntropy \leftarrow totalEntropy - p_i \times \log_2(p_i)$ ▷ Calculate entropy contribution of y_i

```
    end for
    return totalEntropy
end function

function CalculateAttributeWeights(totalEntropy, Y)
    attributeWeights ← []
    for y_i in Y do
        p_i ← Σ_{j=1}^{|Y|} 1(y_i = y_j)          ▷ Number of instances of y_i in Y
        weight_i ← (p_i × log_2(p_i)) / totalEntropy          ▷ Calculate attribute weight
        Append weight_i to attributeWeights
    end for
    return attributeWeights
end function

function F1Score(TP, FP, FN)
    precision ← TP / (TP+FP)
    recall ← TP / (TP+FN)
    return (2·precision·recall) / (precision+recall)
end function
```

Algorithm 2 Categorical Zero CAP

Input: Key attributes A_K, Sensitive attributes A_S, Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$

Output: A risk scoring in the range $[0, 1]$

```
P_corr = {}
for each y in Y do
    T_eq ← from Z, matching y on A_K
    corr ← {z|z ∈ T_eq ∧ y[A_S] = z[A_S]}
    p ← 0
    if T_eq ≠ ∅ then
        p ← |corr| / |T_eq|
    end if
    P_corr ← P_corr ∪ p
end for
return 1− avg(P_corr)
```

Algorithm 2 shows the way Categorical Zero CAP works. For each sensitive tuple y, the algorithm finds the set of anonymised tuples T_{eq} in the equivalence class of that sensitive tuple, i.e. matching on the key attributes A_K, as shown in line 3. For each of these equivalence classes, it calculates the probability p of guessing the sensitive tuple's sensitive attributes correctly and saves this; if T_{eq} is empty, then the algorithm sets p to 0. Lastly, on line 11, it returns the average of these probabilities.

Algorithm 3 Categorical Generalized CAP

Input: Key attributes A_K, Sensitive attributes A_S, Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$

Output: A risk scoring in the range $[0, 1]$

1: $P_{corr} = \{\}$
2: **for** each y in Y **do**
3: $T_{eq} = \emptyset$
4: $\mu = |A_K|$
5: **while** $T_{eq} = \emptyset$ **do** ▷ Keep looking for more distant matches until found
6: $T_{eq} \leftarrow$ from Z, matching y on A, where $A \subseteq A_K$ and $|A| = \mu$ ▷ match on μ attributes
7: $corr \leftarrow \{z | z \in T_{eq} \wedge y[A_S] = z[A_S]\}$
8: **if** $T_{eq} \neq \emptyset$ **then** ▷ If matches found: Save result
9: $p \leftarrow \frac{|corr|}{|T_{eq}|}$
10: $P_{corr} \leftarrow P_{corr} \cup p$
11: **else** ▷ Else: Try matching on 1 fewer attribute
12: $\mu \leftarrow \mu - 1$
13: **end if**
14: **end while**
15: **end for**
16: **return** $1-$ AVG(P_{corr})

Categorical Generalized CAP, described in Algorithm 3 is similar to CZeroCAP. The only difference is how it handles when $T_{eq} = \emptyset$, where instead of setting $p = 0$, it tries to match on one fewer of the key attributes until $T_{eq} \neq \emptyset$. This change is implemented with the value μ on line 4 defining how many attributes need to match, which is used in line 6, and if no matches are found, is decremented on line 12.

5.2 Re-identification metric algorithms

Re-identification metrics revolve around determining which tuple in the anonymised dataset is a given individual in the external knowledge (subset of the sensitive dataset) dataset. An overview re-identification metrics can be found in Table 3, after which the metrics are individually described.

Metric	Type of metric	Applicable attribute types	Additional inputs required	Computation time
Authenticity[4] [9]	Distance	Any	-	Texas: 1055s ± 31 MedOnc: 0.66s ± 0.02
Chi Squared Test* [9]	Statistical	Any; Works best with categorical	Attributes A	Texas: 4.54s ± 0.46 MedOnc: 0.7s ± 0.03
Close Values Probability* [9]	Statistical	Any; Works best with numerical	-	Texas: 4.28s ± 0.40 MedOnc: 0.65s ± 0.01
Distant Value Probability* [9]	Statistical	Any; Works best with numerical	-	Texas: 4.33s ± 0.49 MedOnc: 0.7s ± 0.02
DetectionMLP* [9]	Probabilistic	Any	-	Texas: 4.33s ± 0.48 MedOnc: 0.66s ± 0.02
Inverse Kullback-Leibler Divergence* [9]	Statistical	Any	-	Texas: 4.18s ± 0.27 MedOnc: 0.72s ± 0.08
Jensen Shannon Distance* [9]	Statistical	Any	-	Texas: 4.28s ± 0.32 MedOnc: 512.6s ± 205
Kolmogorov-Smirnov Test* [9]	Statistical	Any	-	Texas: 4.51s ± 0.58 MedOnc: 0.66s ± 0
Common Rows Proportion [9]	Statistical	Any; Works best with categorical and binary	-	Texas: 0.26s ± 0.01 MedOnc: 0.05s ± 0
Identifiability Score [25]	Distance	Any	-	Texas: 0.20s ± 0.01 MedOnc: 0.05s ± 0
NNDR (Nearest Neighbors Distance Ratio)[4] [26]	Distance	Any	-	Texas: 1.37s ± 0.14 MedOnc: 3.83s ± 1.15
NSND (Nearest Synthetic Neighbor Distance)[4] [9]	Distance	Any	-	Texas: 0.99s ± 0.03 MedOnc: 0.07s ± 0
DCR (mean Distance to the Closest Record)[4,5] [26]	Distance	Any	Attributes A	Texas: 2.06s ± 0.28 MedOnc: 1.96s ± 0.03
MDCR (Median Distance to Closest Record)* [12]	Distance	Any (numeric simpler)	-	Texas: 211s ± 6.30 MedOnc: 12.77s ± 0.27

Table 3. Re-identification metrics used to conduct experiments. These are based upon the findings from the review performed in our previous work [7], as well as metrics found in further examination during the making of this paper marked with *

Algorithm 4 functions by first computing the distances to the closest neighbours in the sensitive dataset Y and vice versa for the synthetic dataset Z. Next the distances between each point and its second nearest neighbour in Y and the nearest neighbour in Z are calculated. Then, the authenticity for each sensitive data point in Y is calculated by comparing the distance to its second nearest neighbour in Y with its distance to its nearest neighbour in Z. Finally, authenticity is scored as the proportion of sensitive data points for which the synthetic neighbour is closer than the second nearest sensitive neighbour.

Algorithm 4 Authenticity

Input: Sensitive dataset $Y = \{y_1, ..., y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

$closestSens \leftarrow \text{KNN}(y_i, Y, A, k = 2)$ for each i from 1 to $|Y|$
$closestSynth \leftarrow \text{KNN}(z_i, Z, A, k = 1)$ for each i from 1 to $|Z|$
$sensToSens \leftarrow \textsc{Dist}(y_i, closestSens[i][2])$ for each i from 1 to $|Y|$ ▷ Distance to second nearest sensitive neighbour
$sensToSynth \leftarrow \textsc{Dist}(z_i, closestSynth[i][1])$ for each i from 1 to $|Z|$ ▷ Distance to nearest synthetic neighbour
$AuthScore \leftarrow 0$
for $i \leftarrow 1$ to $|Y|$ **do**
    if $sensToSens[i] < sensToSynth[i]$ **then**
        $AuthScore \leftarrow AuthScore + 1$
    end if
end for
return $\frac{AuthScore}{|Y|}$

Algorithm 5 operates by iterating through each attribute in the sensitive dataset Y. For each attribute, it computes the observed frequencies of each attribute category in both Y and the anonymised dataset Z, as detailed in lines 3 and 4. The algorithm then calculates the total count of frequencies for both datasets. Within a nested loop, it iterates through each category of the attribute, computing the expected frequency and updating the chi-squared statistic accordingly (lines 7-10). After computing the chi-squared statistic for all attribute categories, it calculates the degrees of freedom and the corresponding p-value using a chi-square distribution function [27]. The computed p-values for each attribute are appended to the p_{values} list (line 13). Finally, the algorithm returns the mean of the computed p-values as the risk scoring metric.

[4]The output of this metric has been reversed ($score = 1 - originalscore$)
[5]The output of this metric has been normalised to the range $[0, 1]$
[6]Assumes each anonymised tuple is generated from a single sensitive tuple

Algorithm 5 ChiSquaredTest

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$, Attributes $A = \{a_1, ..., a_m\}$

Output: A risk scoring in the range $[0, 1]$

1: $p_{values} \leftarrow []$
2: **for** i from 1 to $|A|$ **do**
3: $obsFreqSens \leftarrow$ Get observed frequencies of all given categories of attribute a_i in Y
4: $obsFreqSynthetic \leftarrow$ Get observed frequencies of all given categories of attribute a_i in Z
5: $totalCount \leftarrow \textsc{Count}(obsFreqSens) + \textsc{Count}(obsFrewSynthetic)$ ▷ Total count of frequencies for sensitive and synthetic datasets
6: $chi \leftarrow 0$
7: **for** j in range $|obsFreqSens|$ **do** ▷ For each category
8: $expFreq \leftarrow \frac{obsFreqSens[j]+obsFreqSynthetic[j]\times totalCount}{totalCount}$ ▷ Calculate the expected frequency
9: $chi \leftarrow chi + \frac{(obsFreqSens[j]-expFreq)^2}{expFreq}$ ▷ Calculate chi square
10: **end for**
11: $df \leftarrow |obsFreqSens| - 1$ ▷ Compute degrees of freedom
12: $pValue \leftarrow \textsc{ChiSquare}(chi, df)$
13: Append $pValue$ to p_{values}
14: **end for**
15: **return** $\textsc{Mean}(p_{values})$

Algorithm 6, functions by first initialising, in line 1 a threshold value at 0.2 to establish the proximity criteria. Iterating through each tuple y in Y, it computes the minimum distance to tuples z in Z (line 3-14). If this distance is within the threshold, it increments a counter variable, $closeValuesCount$. Finally, it returns the ratio of $closeValuesCount$ to the total number of tuples in Y (line 15).

Algorithm 6 Close Values Probability

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$

Output: A risk scoring in the range $[0, 1]$

$threshold \leftarrow 0.2$
$closeValuesCount \leftarrow 0$
for y in Y **do**
    $minDistance \leftarrow \infty$
    for z in Z **do**
        $distance \leftarrow \textsc{Distance}(y, z)$
        if $distance < minDistance$ **then**
            $minDistance \leftarrow distance$
        end if
    end for
    if $minDistance \leq threshold$ **then**
        $closeValuesCount \leftarrow closeValuesCount + 1$
    end if
end for
return $\frac{closeValuesCount}{|Y|}$

Algorithm 7 Distant Values Probability

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$

Output: A risk scoring in the range $[0, 1]$

$threshold \leftarrow 0.8$
$distantValuesCount \leftarrow 0$
for y in Y **do**
    $minDistance \leftarrow \infty$
    for z in Z **do**
        $distance \leftarrow \textsc{Distance}(y, z)$
        if $distance < minDistance$ **then**
            $minDistance \leftarrow distance$
        end if
    end for
    if $minDistance \geq threshold$ **then**
        $distantValuesCount \leftarrow distantValuesCount + 1$
    end if
end for
return $1 - \frac{distantValuesCount}{|Y|}$

Algorithm 7 functions by assessing the dissimilarity between tuples in the sensitive dataset Y and their farthest counterparts in the anonymised dataset Z. A threshold value of 0.8 is defined in line 1 to set the remoteness criteria and tracks the number of tuples in Y with distant matches in Z. The algorithm iterates through each tuple y in Y, calculating the minimum distance to tuples z in Z (lines 3-14). If this distance exceeds the threshold, it increments a counter variable (lines 7-9). Finally, in line 15 the ratio of the counter variable to the total number of tuples in Y is returned as the risk scoring metric.

Algorithm 8 functions as an MLP feedforward neural network. First we define the labels, after which from line 3-5, Y and Z are gathered both in terms of the raw data and labels, in order to split all the data into separate folds i.e. multiple sets of training and tests sets. Next, from line 7 to 12, the folds are iterated on, where a MLP classifier is trained on the training data. Next the classifier is employed on the test data, whereafter area under curve can be calculated. Finally, in line 13, the average AUC score is returned.

Algorithm 8 DetectionMLP

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$, Number of folds k

1: $L_Y \leftarrow [1, ..., 1]$ of length $|Y|$ ▷ Sensitive dataset labels
2: $L_Z \leftarrow [0, ..., 0]$ of length $|Z|$ ▷ Anonymised dataset labels
3: $D \leftarrow Y \cup Z$
4: $L \leftarrow L_Y \cup L_Z$
5: Split D and L into K-folds i.e. K train and test sets ▷ Number of folds is k
6: $AUCS \leftarrow []$
7: **for** $(D_{train}, D_{test}, L_{train}, L_{test})$) in K-folds **do**
8: Train MLP classifier C on D_{train} and L_{train}
9: $P \leftarrow \mathrm{C}(D_{test})$
10: $AUC \leftarrow \mathrm{ROCAUC}(L_{test}, P)$
11: Append AUC to $AUCS$
12: **end for**
13: **return** AVG($AUCS$)

14: **function** ROCAUC(L_{test}, P)
15: $N_{pos} \leftarrow \mathrm{count}(L_{test} = 1)$ ▷ Number of positive samples
16: $N_{neg} \leftarrow \mathrm{count}(L_{test} = 0)$ ▷ Number of negative samples
17: Sort P in descending order, with corresponding L_{test}
18: $TPR \leftarrow [0]$, $FPR \leftarrow [0]$
19: $TP \leftarrow 0$, $FP \leftarrow 0$
20: **for** each $(label, score)$ in sorted (L_{test}, P) **do**
21: **if** label = 1 **then**
22: $TP \leftarrow TP + 1$
23: **else**
24: $FP \leftarrow FP + 1$

25: **end if**
26: Append $\frac{TP}{N_{pos}}$ to TPR
27: Append $\frac{FP}{N_{neg}}$ to FPR
28: **end for**
29: Append 1 to TPR and FPR
30: $AUC \leftarrow 0$
31: **for** i from 1 to length of TPR **do**
32: $AUC \leftarrow AUC + \frac{(FPR[i]-FPR[i-1])\times(TPR[i]+TPR[i-1])}{2}$
33: **end for**
34: **return** AUC
35: **end function**

Algorithm 9 mainly works by computing inverse KLD (Kullback-Liebler Divergence) [23] for each Y and Z attribute, individually. The score is then the average of these individual inverse KLDs, as seen in line 9. Overall, the metric functions as a means to check for similarity between the two datasets.

Algorithm 9 Inverse Kullback-Leibler Divergence

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

1: $freqs \leftarrow$ Get frequency histograms for Y and Z
2: $invKLs \leftarrow []$
3: **for** a in A **do**
4: $Y_{freq}, Z_{freq} \leftarrow freqs[a]$ ▷ Get frequency distribution for the given attribute a
5: $KL \leftarrow KL_{div}(Y_{freq}, Z_{freq})$ ▷ Calculate KLD for the sensitive and anonymised attributes
6: $invKL \leftarrow \frac{1}{1+KL}$ ▷ Get inverse KLD
7: Append $invKL$ to $invKLs$
8: **end for**
9: **return** AVG($invKLs$) ▷ Get average inverse KLD

Algorithm 10 functions much like Algorithm 9 by employing a similarity measurement on a given Y and Z attribute individually. In this case Jensen-Shannon distance [28] is used as seen on line 5, which utilises KLD to measure the similarity between two probability distributions. The Jensen-Shannon distance works by first calculating the average distribution $M = \frac{1}{2}(P + Q)$ of the two input distributions/attributes, P (from dataset Y) and Q (from dataset Z). It then computes the KLD of each distribution with respect to M. The Jensen-Shannon divergence is the average of these two KL divergences. This is done for each attribute, where finally, on line 8, the average Jensen-Shannon distance is computed.

Algorithm 10 Jensen Shannon Distance

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

1: $freqs \leftarrow$ Get frequency histograms for Y and Z
2: $dists \leftarrow []$
3: **for** a in A **do**
4: $\quad Y_{freq}, Z_{freq} \leftarrow freqs[a]$ ▷ Get frequency distribution for the given attribute a
5: $\quad jsd \leftarrow$ JensenShannonDistance(Y_{freq}, Z_{freq}) ▷ Compute Jensen-Shannon distance on a given attribute a
6: $\quad$ Append jsd to $dists$
7: **end for**
8: **return** $1-$ avg$(dists)$

Algorithm 11 functions much like Algorithm 9 and 10 in that a similarity measurement is employed for attributes separately. In this case it is KST (Kolmogorow-Smirnow Test), which specifically is employed in line 3. Here, KST quantifies the maximum vertical distance between the ECDFs (Empirical Cumulative Distribution Functions) of the attribute a in Y and Z. In line 4 the resulting KST value is subtracted from 1 as to indicate greater similarity between the distributions the higher the value. Finally, the average of the similarity scores for all attributes is computed in line 6.

Algorithm 11 Kolmogorov-Smirnow Test

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

1: $res \leftarrow []$
2: **for** a in A **do**
3: $\quad ks \leftarrow$ ksTest$(Y[a], Z[a])$ ▷ Compute Kolmogorov-Smirnov test statistic for attribute a for each datasets
4: $\quad$ Append $(1 - ks)$ to res
5: **end for**
6: **return** avg(res)

7: **function** ksTest$(sensColumn, anonColumn)$
8: $\quad$ Sort $sensColumn$ and $anonColumn$ in ascending order ▷ Construct ECDFs for both columns
9: $\quad allColumn \leftarrow sensColumn$ concatenated with $anonColumn$
10: $\quad F_{\text{sens}} = \frac{\text{getInsIdx}(sensColumn[i], allColumn)}{|sensAttribute|}$ for each i from 1 to $|sensColumn|$
11: $\quad F_{\text{anon}} = \frac{\text{getInsIdx}(anonColumn[i], allColumn)}{|anonAttribute|}$ for each i from 1 to $|anonColumn|$
12: $\quad$ **return** $D = \max_i (F_{\text{sens}}[i] - F_{\text{anon}}[i])$ ▷ Maximum vertical distance between the two ECDFs
13: **end function**

```
14: function GETSINSIDX(toIns, insList)
15:     for i from 1 to |insList| do
16:         if insList[i] ≥ toIns then
17:             return i
18:         end if
19:     end for
20: end function
```

Algorithm 12 functions simply by performing an intersection check between Y and Z, meaning the tuples they have in common. The resulting score is the proportion of tuples relative to the total number of tuples in Y. Additionally, 1×10^{-8} is added as to prevent division by zero.

Algorithm 12 Common Rows Proportion

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

```
1: intersection ← Y ∩ Z
2: return |intersection| / (|Y|+1×10^−8)        ▷ 1 × 10^−8 is added to avoid division by zero
```

Algorithm 13 Identifiability Score

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

```
                                                ▷ Find nearest neighbours for sensitive and anon datasets
1: NN_sens ← KNN(y_i, Y, A, k = 2) for each i from 1 to |Y|   ▷ Compute 2 nearest neighbours for each data point in Y
2: NN_anon ← KNN(z_i, Z, A, k = 1) for each i from 1 to |Z|   ▷ Compute 1 nearest neighbour for each data point in Z
                                                ▷ Compute re-identification score
3: diff ← {}
4: for i ← 1 to |Y| do
5:     dists_sens ← []
6:     for j ← 1 to 2 do                        ▷ Loop over the 2 nearest neighbours of data point i in Y
7:         dists_sens[j] ← DIST(y_i, NN_sens[i][j])   ▷ Compute distance between y_i and its j-th nearest neighbor
8:     end for
9:     dists_anon ← DIST(z_i, NN_anon[i][1])    ▷ Compute distance between z_i and its nearest neighbour
10:    diff[i] ← dists_anon − dists_sens[2]
11: end for
12: return SUM(diff<0) / n_1                    ▷ Sum over number of instances where diff < 0
```

Algorithm 13 provides an overview of the Identifiability Score, which quantifies the re-identification risk of individuals in Z based on their nearest neighbours in both datasets. In line 1 and 2 KNN is employed to find the nearest neighbours of each datasets. Next, from line 4-11 the re-identification score is computed. For each i from 0 to $|Y|$, the distance

between y_i and its 2nd nearest neighbour is computed as well as the distance between z_i and its nearest neighbour. Using this approach, it is determined whether the nearest neighbour in Z is closer to z_i than the 2nd nearest neighbour in Y, which could indicate a risk of re-identification. Finally, in line 12, the resulting Identifiability Score is computed as the division of the number of instances where the nearest neighbour in Z is closer than the 2nd nearest neighbour in Y, by the total number of tuples in Y.

The way NNDR works is described in Algorithm 14. The first step here is to project both datasets down to two numeric dimensions [29]. After this, inside the loop on line 5 it computes the closest and second-closest sensitive tuple for each anonymised tuple. Using this, it computes for each anonymised tuple the ratio between these two distances; if both have a distance of zero we define the ratio as 1. Lastly, these ratios are averaged and the result is 1 minus this average, to make more private data give a lower score.

Algorithm 14 NNDR

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

1: $coord_Y \leftarrow$ PROJECT$(Y, n_dim = 2)$ ▷ Project data down to 2 numeric dimensions
2: $coord_Z \leftarrow$ PROJECT$(Z, n_dim = 2)$
3:
4: $ratios \leftarrow []$
5: **for** i from 1 to n_2 **do** ▷ for each anonymised tuple, find the 1st and 2nd closest sensitive tuple
6: $NNs \leftarrow$ KNN$(coord_Z[i], coord_Y, A, k = 2)$
7: $dist_1 \leftarrow NNs[1]$
8: $dist_2 \leftarrow NNs[2]$
9: **if** $dist_2 = 0$ **then** ▷ avoid division by zero
10: Append 1 to $ratios$
11: **else**
12: Append $\frac{dist_1}{dist_2}$ to $ratios$
13: **end if**
14: **end for**
15: **return** $1-$ AVG$(ratios)$

Similar to NNDR is NSND, as described in Algorithm 15. The key difference is that instead of it being a distance ratio, it is more simply a distance from each sensitive tuple to its closest anonymised tuple, as is calculated in the loop on line 2. Because it isn't a ratio, it is normalised to the range $[0, 1]$ in the loop on line 8, after which the distances are again averaged and it returns 1 minus this average for the same reason as before.

Algorithm 15 NSND

Input: Real dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$
Output: A risk scoring in the range $[0, 1]$

1: $dists \leftarrow []$
2: **for** $y \in Y$ **do** ▷ Compute distances for each sensitive tuple to its nearest anonymised tuple
3: $NNs \leftarrow \text{KNN}(y, Y, A, k = 1)$
4: $closest\_dist \leftarrow NNs[1]$
5: Append $closest\_dist$ to $dists$
6: **end for**
7: $dists\_norm \leftarrow []$
8: **for** i from 1 to $|dists|$ **do** ▷ Normalise to [0,1]
9: $dist\_norm \leftarrow \frac{dists[i]-min(dists)}{(max(dists)-min(dists))+1\times10^{-8}}$ ▷ 1×10^{-8} is added to avoid division by zero
10: Append $dist\_norm$ to $dists\_norm$
11: **end for**
12: **return** 1− AVG($dists\_norm$)

In Algorithm 16 the DCR metric is described. Firstly, it projects the datasets down to two numeric dimensions. Then, on line 5 and inside the loop, it finds the closest fake coordinate for each sensitive coordinate. These distances are then averaged, and the result is as shown on line 10, which uses the *sigmoid* and *log* functions.

Algorithm 16 DCR

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$, Attributes $A = \{a_1, ..., a_m\}$
Output: A risk scoring in the range $[0, 1]$

1: $coord_Y \leftarrow \text{PROJECT}(Y, n\_dim = 2)$ ▷ Project data down to 2 numeric dimensions
2: $coord_Z \leftarrow \text{PROJECT}(Z, n\_dim = 2)$
3:
4: $dist\_arr \leftarrow []$
5: **for** i from 1 to $|Y|$ **do** ▷ For each sensitive coord, find closest fake coord
6: $dist \leftarrow min_j(|coord_Y[i] - coord_Z[j]|)$
7: Append $dist$ to $dist\_arr$
8: **end for**
9: $avg = \text{AVG}(dist\_arr)$
10: **return** 1− SIGMOID(LOG(avg))

In Algorithm 17, MDCR works similarly to DCR (in Algorithm 16), though with some differences: (1) it does not use projection for the data, (2) it also calculates the distances within-dataset (on line 5) and (3) the final calculation differs, including using *median* instead of *avg*.

Algorithm 17 MDCR

Input: Sensitive dataset $Y = \{y_1, ...y_n\}$, Anonymised dataset $Z = \{z_1, ..., z_n\}$

Output: A risk scoring in the range $[0, 1]$

$betweens = []$
$withins = []$
for i from 1 to n **do** ▷ Calc. the min. distances between- and within-dataset
    Append $min_j(|z_i - y_j|)$ to $betweens$
    Append $min_{j,j\neq i}(|y_i - y_j|))$ to $withins$
end for
$medi = \frac{\text{MEDIAN}(betweens)}{\text{MEDIAN}(withins)}$
return $3 - \text{SIGMOID}(medi)$

5.3 Tracing metric algorithms

The last category of metrics considered is tracing metrics, which attempt to infer whether a given tuple in the external knowledge (subset of the sensitive dataset) dataset was part of the dataset used to create the synthetic dataset. The overview for these can be seen in Table 4, after which again the metrics are described individually.

Metric	Type of metric	Applicable attribute types	Additional inputs required	Computation time
NNAA (Nearest neighbour adversarial accuracy) [12]	Distance-based accuracy ratio	Any (numeric simpler)	-	Texas: $406s \pm 7.24$ MedOnc: $27s \pm 0.17$
Membership Inference Risk [12]	Distance-based accuracy	Any (numeric simpler)	distance threshold dt	Texas: $16.59s \pm 3.32$ MedOnc: $26.91s \pm 0.49$
Hidden Rate[7,8,9] [26]	KNN accuracy[9]	Any	-	Texas: $1.74s \pm 0.09$ MedOnc: 2.04 ± 0.03
Hitting Rate* [12]	Duplicate scan	Any	Attributes A	Texas: $149s \pm 5.61$ MedOnc: $15.39s \pm 0.24$

Table 4. Tracing metrics used to conduct experiments. These are based upon the findings from the review performed in our previous work [7], as well as metrics found in further examination during the making of this paper marked with *

[7]The output of this metric has been reversed ($score = 1 - originalscore$)
[8]The output of this metric has been normalised to the range $[0, 1]$
[9]Assumes each anonymised tuple is generated from a single sensitive tuple

The pseudocode for NNAA is in Algorithm 18. It uses a distance function as defined on line 1, which finds the minimum distance from tuple i in dataset D_1 to any tuple in D_2; if the same datasets are given as input, the second-closest tuple is used (i.e. not checking the tuple against itself). On line 14, we go through each index in Y and Z, and for each of these datasets, we keep count of the number of times it is more distant (i.e. less accurate) to use the opposite dataset than it is to use the same dataset. These values are then combined and averaged on line 22, giving a sort of ratio of the accuracy between-dataset against within-dataset. Lastly the value is inverted so lower values indicate more private data (lower accuracy between-dataset), which happens on line 25.

Algorithm 18 NNAA[10]

Input: Sensitive dataset $Y = \{y_1, ...y_n\}$, Anonymised dataset $Z = \{z_1, ..., z_n\}$
Output: A risk scoring in the range $[0, 1]$

1: **function** DIST(D_1, D_2, i) ▷ Returns min. dist. from tuple i in D_1 to any tuple in D_2
2: $\{r_1, ..., r_n\} \leftarrow D_1$
3: $\{t_1, ..., t_n\} \leftarrow D_2$
4: **if** $D_1 = D_2$ **then** ▷ Avoid checking tuple against itself
5: **return** $min_{j,j\neq i}|r_i - t_j|$ ▷ distance between r_i and t_j
6: **else**
7: **return** $min_j|r_i - t_j|$
8: **end if**
9: **end function**
10:
11: **function** $AA_{YZ}(Y, Z)$
12: $Sum_{YZ} \leftarrow 0$
13: $Sum_{ZY} \leftarrow 0$
14: **for** $i \leftarrow 1$ to n **do** ▷ Sum no. of times between-dataset is more distant than within-dataset
15: **if** DIST$(Y, Z, i) >$ DIST(Y, Y, i) **then**
16: $Sum_{YZ} \leftarrow Sum_{YZ} + 1$
17: **end if**
18: **if** DIST$(Z, Y, i) >$ DIST(Z, Z, i) **then**
19: $Sum_{ZY} \leftarrow Sum_{ZY} + 1$
20: **end if**
21: **end for**
22: **return** $\frac{1}{2\cdot|Z|} \cdot (Sum_{YZ} + Sum_{ZY})$
23: **end function**
24:
25: **return** $1- AA_{YZ}(Y, Z)$

[10]The original metric includes a holdout dataset, but the implementation by Lautrup et al. [12] includes a version without a holdout, which is the one used.

Membership Inference Risk works similarly to NNAA (Algorithm 18), in that it also uses distance to calculate an accuracy, but is significantly simpler and can be seen in Algorithm 19. On line 2 it goes through each sensitive sample; based on the closest anonymised sample to this, in the next two lines it guesses whether the sensitive sample was used to generate the anonymised dataset. With this it counts up the true positives TP and false negative FN, which is used on line 10 to calculate the recall of these guesses, which is then the value returned.

Algorithm 19 Membership Inference Risk[11]

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$, distance threshold dt
Output: A risk scoring in the range $[0, 1]$

1: $TP, FN \leftarrow 0, 0$ ▷ True/False, Positives/Negatives (for recall)
2: **for** $y \in Y$ **do** ▷ Is a true sample
3: $dist_{min} \leftarrow min_j|y - z_j|$ ▷ Minimum Euclidean distance
4: **if** $dist_{min} \leq dt$ **then** ▷ Guesses true sample
5: $TP \leftarrow TP + 1$
6: **else** ▷ Guesses false sample
7: $FN \leftarrow FN + 1$
8: **end if**
9: **end for**
10: $recall \leftarrow \frac{TP}{TP+FN}$
11: **return** $recall$

Next, Algorithm 20 shows how Hidden Rate works. First, it projects the datasets down to two numeric dimensions. It then uses these coordinates on line 6 for each sensitive sample, to predict which fake sample was based on this. On the next line it then checks if this prediction is correct, and if correct it counts *n_closest* up by one. Lastly it returns the ratio of samples guessed correctly.

[11] The metric from Lautrup et al. [12] is based on an F1 score, but includes a version based on recall, which is the one used.

Algorithm 20 Hidden Rate

Input: Sensitive dataset $Y = \{y_1, ...y_n\}$, Anonymised dataset $Z = \{z_1, ..., z_n\}$
Output: A risk scoring in the range $[0, 1]$

1: $coord_Y \leftarrow \text{PROJECT}(Y, n\_dim = 2)$ ▷ Project data down to 2 numeric dimensions
2: $coord_Z \leftarrow \text{PROJECT}(Z, n\_dim = 2)$
3:
4: $n\_closest \leftarrow 0$
5: **for** i from 0 to $|Y|$ **do**
6: $NNs \leftarrow \text{KNN}(coord_Y[i], coord_Z, A, k = 1)$ ▷ Predict with KNN
7: **if** $NNs[1] = i$ **then**
8: $n\_closest \leftarrow n\_closest + 1$
9: **end if**
10: **end for**
11: ▷ Check indices and count up $n\_closest$
12: **return** $\frac{n\_closest}{n}$

Lastly, Hitting Rate is described in Algorithm 21. On line 1, the *threshold* is set to $\frac{100\%}{30}$ of the attribute's sensitive-data range. What is meant by this is that for whichever attribute a this *threshold* is applied to, the *threshold* will be equal to $(max_i(y_i[a]) - min_i(y_i[a])) \cdot \frac{1}{30}$, which is roughly 3% of that attribute's total range in the sensitive data. Next, the attributes are split into the numerical and categorical attributes. these are then used on line 4, where the subset of Z that "hits" some sensitive tuple is calculated; In order to "hit" a tuple, it needs to exactly match on the categorical attributes, and be within the *threshold* of that tuple for each numerical attribute. lastly, the ratio of *hits* to $|Y|$ is returned.

Algorithm 21 Hitting Rate

Input: Sensitive dataset $Y = \{y_1, ...y_{n_1}\}$, Anonymised dataset $Z = \{z_1, ..., z_{n_2}\}$, Attributes $A = \{a_1, ..., a_m\}$
Output: A risk scoring in the range $[0, 1]$

1: $threshold \leftarrow \frac{100\%}{30}$ of the attribute's sensitive-data range
2: $A_n \leftarrow$ numerical attributes of A
3: $A_c \leftarrow$ categorical (and binary) attributes of A
4: $hits \leftarrow$ from Z, where some y matches on A_c and is within *threshold* on A_n
5: **return** $\frac{|hits|}{|Y|}$

6 EXPERIMENTS

In this section, we describe the experiments, including the setup thereof, what the objectives are, and what our hypothesis is for each experiment. These relate to the problems defined in Section 3, with an outlook on answering the questions posed.

6.1 Datasets

For these experiments we use two datasets: MedOnc (Medical Oncology) [30] and Texas[12] [31]. These are the sensitive datasets, with the anonymised datasets being generated based on these, as will be described in Section 6.3.

The MedOnc dataset comprises medical oncology records from patients diagnosed with malignant tumors and treated at Aalborg University Hospital's Department of Oncology. The file we use from this is medonc_patient_multi_measures.csv, which has a total of 162,411 tuples, though we use all tuples without NA values, which consists of 8,630 tuples.

The Texas Hospital Inpatient Discharge Public Use Data (Texas dataset, File 2013 Q1-Q4) is a public dataset from the Texas Department of State Health Services, Austin, Texas. The full dataset contains 100,000 tuples with 6 numerical attributes and 12 categorical attributes. The part of the dataset we used contains 25,000 samples uniformly sampled from a pre-processed dataset[12] [16]. The reason for choosing 25,000 samples is due to computation time limitations.

A summary of the datasets is in Table 5.

Dataset	Full dataset tuples	Tuples used	Columns
Texas	100,000	25,000	18
MedOnc	162,411	8,639	11

Table 5. Datasets summary

For each of the datasets, three separate runs are conducted, which entails generating three different anonymised datasets for a single synthesiser, and then running all metrics on each of the three generated datasets. This is done to also calculate deviation, with some confidence in what that deviation is. In some cases the deviance in the results may be high - in these cases we continue with additional runs until the deviance is low enough. The reason this decreases deviance is that we use SEM (Standard Error of the Mean) [32] to calculate the deviance, which signifies the uncertainty in the true mean of the results.

[12] via https://github.com/spring-epfl/synthetic_data_release, accessed 12-03-2024. In our code the first 25,000 rows are chosen, in order of the file sequence, which is effectively also uniformly distributed.

6.2 Hardware

The hardware used for running the experiments differs relative to the datasets. This is mainly due to MedOnc containing private health data, which we are not allowed to store locally. Here, the high-performance computing cloud environment UCloud[13] is used allowing us to perform the experiments through their infrastructure.

For Texas the experiments were simply run on a local machine.

An overview of the hardware used can be seen in Table 6

Table 6. Hardware specifications used for experiments

Hardware	Processor model	# of Threads	Graphics Card	RAM
UCloud	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz	16	None	96GB
Local	Intel(R) Core(TM) i7-9750H CPU @ 2.50GHz	12	NVIDIA GeForce GTX 1660 Ti	16GB

6.3 Synthesisers

Aside from baselines, four synthesisers to generate data are used to conduct the experiments. Multiple synthesisers are used as to be able to generalise the results and not be dependent on one given synthesiser. These are separated into privacy- and non-privacy-focused synthesisers. Privacy-focused synthesisers are defined as having a type of explicit privacy mechanism incorporated into their framework, whereas non-privacy-focused synthesisers do not have such a mechanism.

Two baselines are used: Real and Random. The Real baseline works as the identity function where the output dataset is identical to the input dataset, consisting of the sensitive data. Real functions as the worst-case scenario where the metric scores are expected to be ~1. In a theoretical point of view, ϵ for Real can be approximated to infinity.

The Random synthesiser generates a random dataset of the same format as the sensitive dataset. Random functions as a best-case scenario with the metric scores being expected to be ~0.

Pseudocode describing Random is provided in Algorithm 22. Generally, this synthesiser only differentiates between categorical and numerical values, where for categorical values there may be introduced more categories, and for numerical values there is extremely high noise. Ideally Random should have an ϵ of zero, though this implementation derives some metadata from the dataset, giving it a non-zero ϵ value, similar to the original implementation of PrivBayes [16]. This metadata for Random is, for each attribute: The set of unique values for that attribute (line 3), and for numerical attributes, the interval (line 6).

[13] https://cloud.sdu.dk

Algorithm 22 Synthesiser: Random

Input: Set of attributes $A = \{a_1, ..., a_m\}$, Sensitive dataset $Y = \{y_1, ..., y_{n_1}\}$

Output: Anonymised dataset $Z = \{z_1, ..., z_{n_1}\}$

$Z \leftarrow Y$
for each a in A **do** ▷ For each attribute, replace all values in Z
  $values \leftarrow \{y[a] \mid y \in Y\}$
  $noise \leftarrow []$ ▷ For each tuple, what noise to add at the end, if numerical
  if a is numerical **then** ▷ Numerical attribute handling
    $interval \leftarrow max_i(y_i[a]) - min_i(y_i[a])$
    $scale, loc \leftarrow interval, 0$
    for $k \leftarrow 1$ to 100 **do**
      $scale \leftarrow scale + max(1,\ n \sim \text{Laplace}(interval, 10 \cdot interval))$
      $loc \leftarrow loc + max(1,\ n \sim \text{Laplace}(interval, 10 \cdot interval))$
    end for
    $noise \leftarrow [ns_1, ..., ns_{n_1}]$ where $ns_1, ..., ns_{n_1} \sim \text{Laplace}(loc, scale)$
  end if
  if a is categorical **then** ▷ Categorical attribute handling
    $nCats \leftarrow |values|$
    $nCatsNew \leftarrow max(100, nCats) + \text{Laplace}(0, max(10, \frac{nCats}{10})))$
    $nCatsNew \leftarrow \lceil max(2, nCatsNew) \rceil$
    if $nCatsNew < nCats$ **then**
      $values \leftarrow Vals \in \{V | V \subset values \wedge |V| = nCatsNew\}$
    else
      for $i \leftarrow 1$ to $(nCatsNew - nCats)$ **do**
        $values \leftarrow values \cup (elem \in \{v | v \in a \wedge v \notin values\})$
      end for
    end if
  end if
  for $j \leftarrow 1$ to n_1 **do** ▷ Generating new data for the attribute
    $z_j \leftarrow elem \in values$
    if $a = \mathbb{R}$ **then**
      $z_j \leftarrow z_j + ns_j$
    end if
  end for
end for
return Z

function Laplace($loc, scale$)
  $f(x) = \frac{1}{2 \cdot scale} \cdot e^{-\frac{|x - loc|}{scale}}$
  return $f(x)$
end function

The privacy-focused synthesisers are: PATE-GAN and PrivBayes. PATE-GAN [19] (Private Aggregation of Teacher Ensembles Generative Adversarial Network) applies the PATE framework [33] on a GAN setup. Due to utilising the PATE framework, PATE-GAN is ensured DP guarantees [34]. PrivBayes [35] employs DP in a Bayesian network, specifically by injecting noise into a set of low dimensional marginals of the sensitive dataset. Then using the noisy marginals and a Bayesian network, an anonymised dataset is generated.

As a note, all privacy-focused techniques incorporate DP. For PrivBayes, delta is fixed: $\delta = 0$, while for PATE-GAN it varies by dataset[14]: For Texas, $\delta = 2.53 \cdot 10^{-7}$, while for MedOnc $\delta = 1.25 \cdot 10^{-6}$.

The non-privacy focused synthesisers are: CTGAN and TVAE. CTGAN [36] (Conditional Tabular Generative Adversarial Network) generates tabular data and also allows specifying certain conditions/constraints in order to generate data with certain features (e.g. generate data for individuals in a specific age bracket). TVAE [9] (Tabular Variational Auto Encoder) is a variational autoencoder focused on generating tabular data.

6.4 Metric sanity test experiments

These experiments revolve around testing whether the metrics from Section 5 work as intended, as specified by Problem (I) in Section 3. This entails that the metrics should output a score relative to the anonymisation level of the anonymised dataset it receives as input. How this is tested is split into two experiments.

6.4.1 Experiment 1.1: Synthesiser-Metric analysis. For this first experiment, the objective is to test whether the outputs of the metrics follow the anonymisation level of a synthesiser. To test this we assume that the Random and Real baselines provide the best and worst level of anonymisation, respectively. All other synthesisers lie between these baselines, no matter the parameters for the technique. If a metric's results do not reflect this, then the metric does not pass the sanity test.

Our hypothesis with this experiment is that the metrics will reflect this ordering of synthesisers in terms of the level of anonymisation. This is because Random has close to no data leakage, while Real has full data leakage, and the anonymisation level should be inversely proportional to this.

6.4.2 Experiment 1.2: Parameter-Metric analysis. In this experiment, we study the correlation between privacy metric scores and θ values. This is done by varying the θ values of a given synthesiser and then applying the metrics. The synthesisers used are PATE-GAN and PrivBayes. These employ DP, therefore having ϵ and δ as parameters. Most relevant for this experiment is the ϵ value, as we want to adjust the privacy budget; The ϵ values used are $[0.2, 1, 2, 10]$. The value of δ is 0 for PrivBayes, while for PATE-GAN it depends on the dataset: for Texas $\delta = 2.53 \cdot 10^{-7}$, while for MedOnc $\delta = 1.25 \cdot 10^{-6}$, as described in Section 6.3.

We hypothesise that anonymised datasets generated with low ϵ values produce better (lower) privacy metric scores, meaning the dataset is more anonymised. With higher ϵ values we presume this to be the opposite. Therefore, a monotonic increase for the metric scores are expected, as we go from lower ϵ values to higher ones.

[14] In the code it is automatically calculated as $\delta = \frac{1}{n*\sqrt{n}}$, where n is the size of the dataset

6.5 Metric selection experiments

These experiments aim to assess whether the metrics capture different aspects of anonymisation, as specified by Problem (II) in Section 3. This is done by evaluating the similarity between individual metrics' outputs, meaning that if two specific metrics have highly similar outputs, it indicates that they capture the same aspects, in terms of anonymisation.

We perform this assessment using two experiments. The data for both of these experiments is all of the experimental data from Experiment 1.1, averaged across runs. This means there will be twelve data points for each metric: The two baselines' values, the two non-privacy-focused synthesisers' values, and four values for each of the two DP-enabled synthesisers. The reason for averaging is to reduce noise, and we use all the data because we want to capture any differences between the metrics, which we expect to be better captured using more synthesisers.

6.5.1 Experiment 2.1: Spearman. The first approach to measure the correlation between the metrics is Spearman Correlation [37]. In our previous work (Hansen et al. [7]) we used Pearson Correlation, but this has been changed to Spearman as this is less influenced by outliers. Spearman Correlation measures the monotonic relation between two variables. The monotonic relationship determines how the scores evolve in relation to each other. With this, a score is produced in the range $[-1, 1]$, where 1 indicates a perfect positive monotonic relationship, where if one score increases, the other does as well. -1 is a perfect negative monotonic relationship whereas one score increases, the other decreases. 0 indicates no monotonic relationship. Spearman Correlation is calculated using Equation 2.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{2}$$

In Equation 2, n is the number of data points and d_i is the difference in the values of a given rank between two privacy metrics. Rank refers to the way Spearman Correlation assigns a position to a given data point. For example in $[2.5, 1, 0, 3]$, the rankings would be $[3, 2, 1, 4]$. This is equivalent to Pearson Correlation for the ranks of the points.

Using Equation 2, the Spearman Correlation is calculated for each pair of metrics.

We hypothesise that metrics of the same privacy attack type are more correlated than those of different attack types.

6.5.2 Experiment 2.2: Clustering. The second approach to measure the correlation between the metrics is clustering. With clustering we cluster the metrics using the density-based OPTICS (Ordering Points To Identify the Clustering Structure) [38] algorithm. OPTICS was chosen as it is an established clustering algorithm with an easily available code implementation. Additionally, it is highly flexible and does not require specifying the number of clusters beforehand.

Generally, clustering involves grouping similar data points according to certain criteria. In OPTICS, a distance measure (e.g., Euclidean distance) is employed to identify the nearest neighbouring data point with the highest density.

We use all the data for this clustering, averaged across runs.

We hypothesise that metrics of the same privacy attack type are clustered together, and we expect there to be three clusters, matching the number of privacy attack types considered.

7 RESULTS

In this section, we present the results of the experiments and an analysis of the implications thereof. We evaluate these results to study the hypothesis, and provide more analysis for the results that were unexpected.

7.1 Experiment 1.1: Synthesiser-Metric analysis

The hypothesis for this experiment is that Random will perform best (lowest scores), Real will perform worst (highest scores), and thereby all other methods will fall inbetween.

First, we discuss the results for the Texas dataset. As expected, in Figure 5 we see that generally, metrics behave as expected: Random gets the best (lowest) metric scores, and Real similarly gets the worst scores, with other values lying inbetween. For Random, the final number of runs was 15, due to the high variance of CloseValueProb, DistantValueProb and NearestSynNeighborDistance. For PATE-GAN, the final number of runs was 5 due to the variance of MDCR (up to 0.18, which is the highest) and MemInf.

There are a few metrics that do not follow this, though. For MDCR, the highest score is for TVAE, and for NearestSynNeighborDistance it is CTGAN that gets the highest score. The Random method does not get the lowest score for ChiSquaredTest and Hidden Rate, for which TVAE and PATE-GAN ($\epsilon = 0.2$) get the lowest scores, respectively. Furthermore, Random has rank-ties for AttrubuteInference, CZeroCAP and CommonRowsProportion. For Real, there is a rank tie on DistantValueProb.

The metrics; CZeroCAP, CommonRowsProportion and IdentifiabilityScore and Hidden_rate all get a score ~ 1 for Real, and 0 for all other synthesisers. This is unexpected as only random is expected to have a score ~ 0 and then all other synthesisers above it to some degree.

The metrics CloseValueProb, ChiSquaredtest, NearestSynNeighbourDistance, DistantValueProb, InvKullbackLieb and JensenShannonDist do not work as expected, where random gets a high score, which is not ~ 0. However, for most of them Random still gets the lowest value. This could indicate an implementation error or that the metrics needs to be tuned for the Texas dataset to work better.

This leaves a total of 13 out of 21 metrics that behave as expected when using the Texas dataset. It is, however, notable that for the exceptions, the results are usually close to each other, meaning that for most of the exceptions, the hypothesis is almost true. The only major exception is ChiSquaredtest, where Random performs measurably worse than CTGAN, TVAE and PrivBayes ($\epsilon = 0.2$). The next-largest exception is the MDCR metric where Real performs measurably better than TVAE. (Measurably referring to outside deviance)

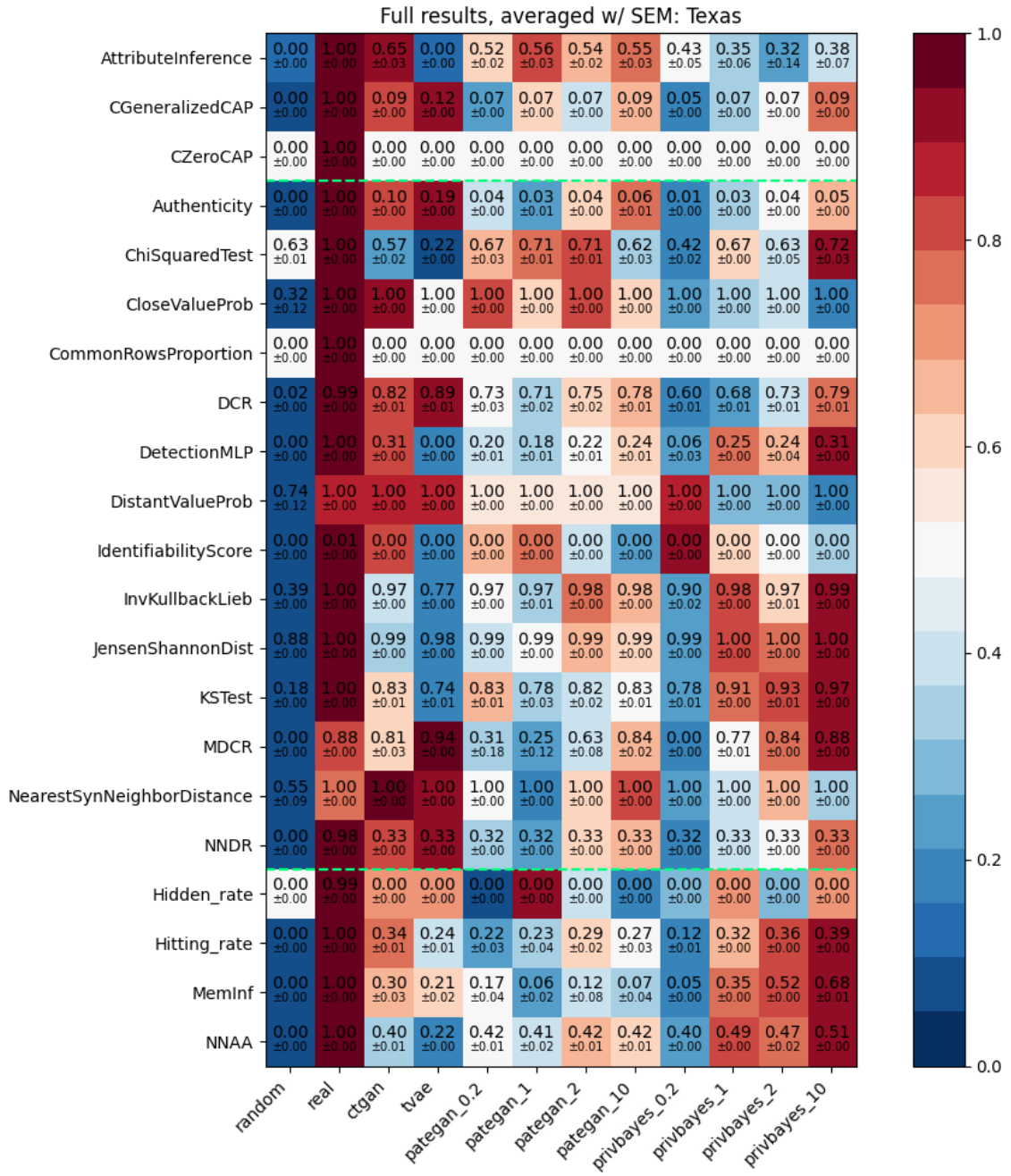

Fig. 5. Rank-derived privacy metric scores on anonymised datasets generated from the Texas dataset. In each cell, the value at the top is the average of the runs, while the value below is the deviance, specifically SEM. For each metric, the methods are ranked from best to worst, and this ranking determines the colour of each cell for that metric, with the highest score (worst) being red, and lowest score (best) being blue. The green lines indicate the boundaries between different attack categories

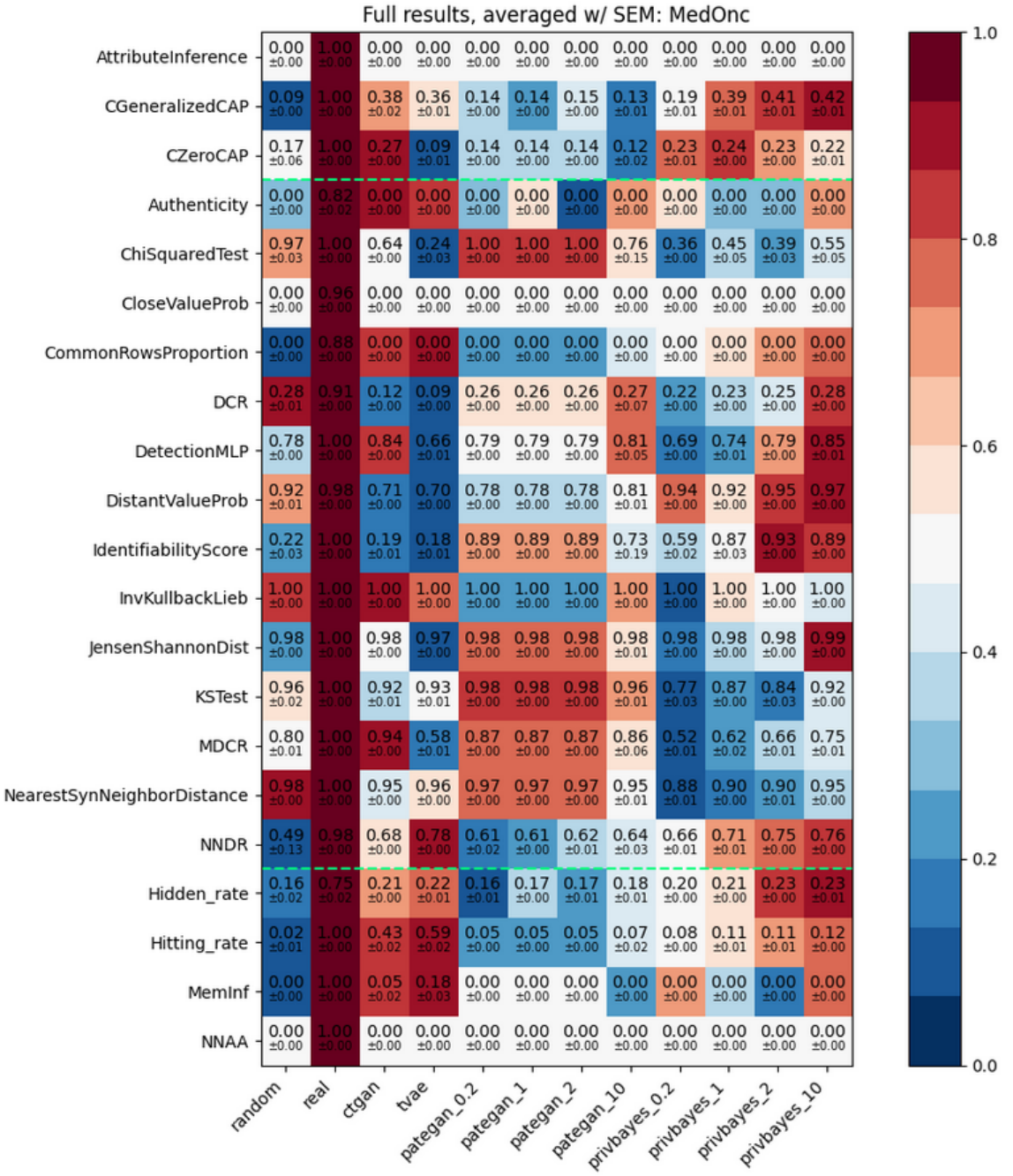

Fig. 6. Rank-derived privacy metric scores on anonymised datasets generated from the MedOnc dataset. In each cell, the value at the top is the average of the runs, while the value below is the deviance, specifically SEM. For each metric, the methods are ranked from best to worst, and this ranking determines the colour of each cell for that metric, with the highest score (worst) being red, and lowest score (best) being blue. The green lines indicate the boundaries between different attack categories

In Figure 6, the results for MedOnc can be seen. Differently from Figure 5, the results here are not as expected with Random only getting the lowest or tied for lowest scores on 8 out of 21 of the metrics with TVAE and PrivBayes $\epsilon = 0.2$ getting similar performance. However, Real functions as expected with it being the worst case scenario (highest score for each metric). The problem with Random could be attributed to the MedOnc dataset not containing as much data as Texas and the Random algorithm itself (discussed in Section 8.3).

An interesting observation is the scores for PATE-GAN $\epsilon = 0.2$, PATE-GAN $\epsilon = 1$ and PATE-GAN $\epsilon = 2$, which are almost identical. PATE-GAN $\epsilon = 10$ gets extremely similar scores, but with slight variance. This pattern can to some degree also be likened to Figure 5. Therefore, it can be suspected that the implementation of PATE-GAN does not function as intended or it could be attributed to MedOnc not having enough data for PATE-GAN to learn and generate varying data. On the other hand, with PrivBayes there is way more variance between the individual configurations.

Also differently from Figure 5, there are more rank-ties, such as AttributeInference, Authenticity, ChiSquaredTest, CloseValueProb, Common Rows Proportion and NNAA. There are also fewer metrics that work as expected for MedOnc, due to Random often not performing well. For Medonc, only 5 of 21 metrics perform as expected due to this.

To summarise, the hypothesis for Experiment 1.1 seems to be mostly true. For Texas, in almost all cases, Random performs best and Real performs worst. For MedOnc, Random has varied results, while Real performs worst in all cases. Furthermore, in most of the cases where this ordering does not hold, the results are close. As a breakdown per-metric, we deem that across the two datasets, the metrics perform according to Table 7.

Works adequately	**Sometimes works**	**Inadequate functionality**
• CGeneralizedCAP	• Attribute Inference	• ChiSquaredTest
• DetectionMLP	• CZeroCAP	• CloseValueProb
• Hitting Rate	• Authenticity	• CommonRowsProportion
• MemInf	• DCR	• DistantValueProb
	• IdentifiabilityScore	• NearestSynNeighborDistance
	• InvKullbackLieb	
	• JensenShannonDist	
	• KSTest	
	• MDCR	
	• NNDR	
	• Hidden Rate	
	• NNAA	

Table 7. Metric functionality breakdown for Experiment 1.1

7.2 Experiment 1.2: Parameter-Metric analysis

Similar to Experiment 1.1, the hypothesis is that for the methods with privacy-adjusting parameters in θ, the metrics' results align with adjustments to these parameters.

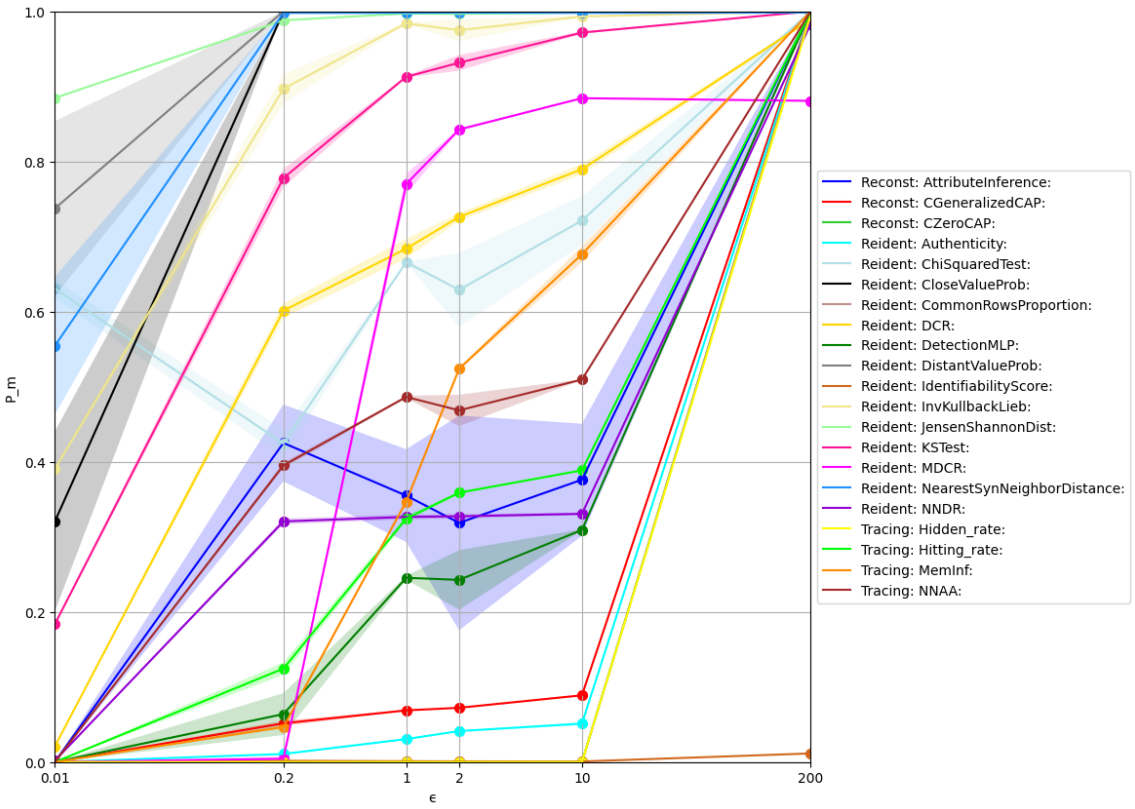

Fig. 7. Metric results for PrivBayes on Texas with different values of epsilon, with logarithmic ϵ axis. The endpoints for ϵ, however, are the Random and Real results, with ϵ set to 0.01 and 200, respectively. The exact choice of these endpoint ϵ values was mostly for visual purposes; theoretical values are discussed in Section 6.3. The deviation is SEM, and is shown with the translucent areas around each line

For PrivBayes on Texas (Figure 7), the Random and Real results have been added to each end of the graph, to further show how the results lie inbetween the baselines. Looking at the graph for this, the results seem to mostly follow the hypothesis, in the sense that all of the metrics increase from one end to the other, and it seems to be mostly monotonic. Only ChiSquaredTest deviates from this past the deviation boundaries, though AttributeInference has higher deviance than than it changes between ϵ values of 0.2 and 10.

This means that 19 of the 21 metrics follow the hypothesis in this test, though there is a great spread in how accurately they capture the further expectation that the greatest change should happen around the middle of this graph. The metrics that are closest to capturing this are DCR and particularly MemInf. A few of the results also have somewhat high deviance, which is suboptimal as it reduces result confidence, with almost half of the metrics showing little to no measurable increase within the middle four values of ϵ.

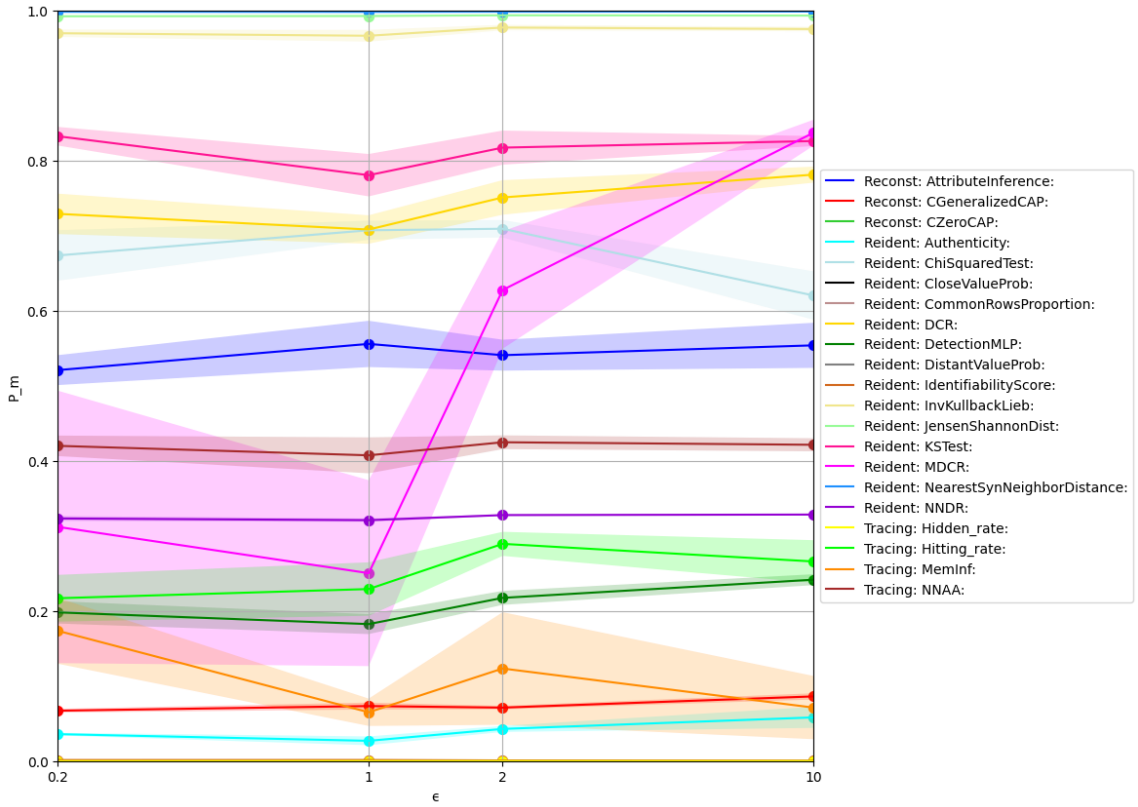

Fig. 8. Metric results for PATE-GAN on Texas with different values of epsilon, with logarithmic ϵ axis. The deviation is SEM, and is shown with the translucent areas around each line

Next, the results for PATE-GAN on Texas (Figure 8) seem much less indicative of the hypothesis. The majority of the metrics show little to no variance between the values of ϵ. The only metric that could indicate an increase outside variance is MDCR, though it has a high SEM. The endpoints of Random and Real are not included in this figure to clarify this lack of variation with changing values of ϵ.

As a summary for the Texas dataset, the hypothesis seems to mostly hold true with the PrivBayes method, with PATE-GAN showing mostly no effect from changing ϵ. Here, also ChiSquaredTest decrease significantly from $\epsilon = 2$ to $\epsilon = 10$. In this case it however also includes DetectionMLP.

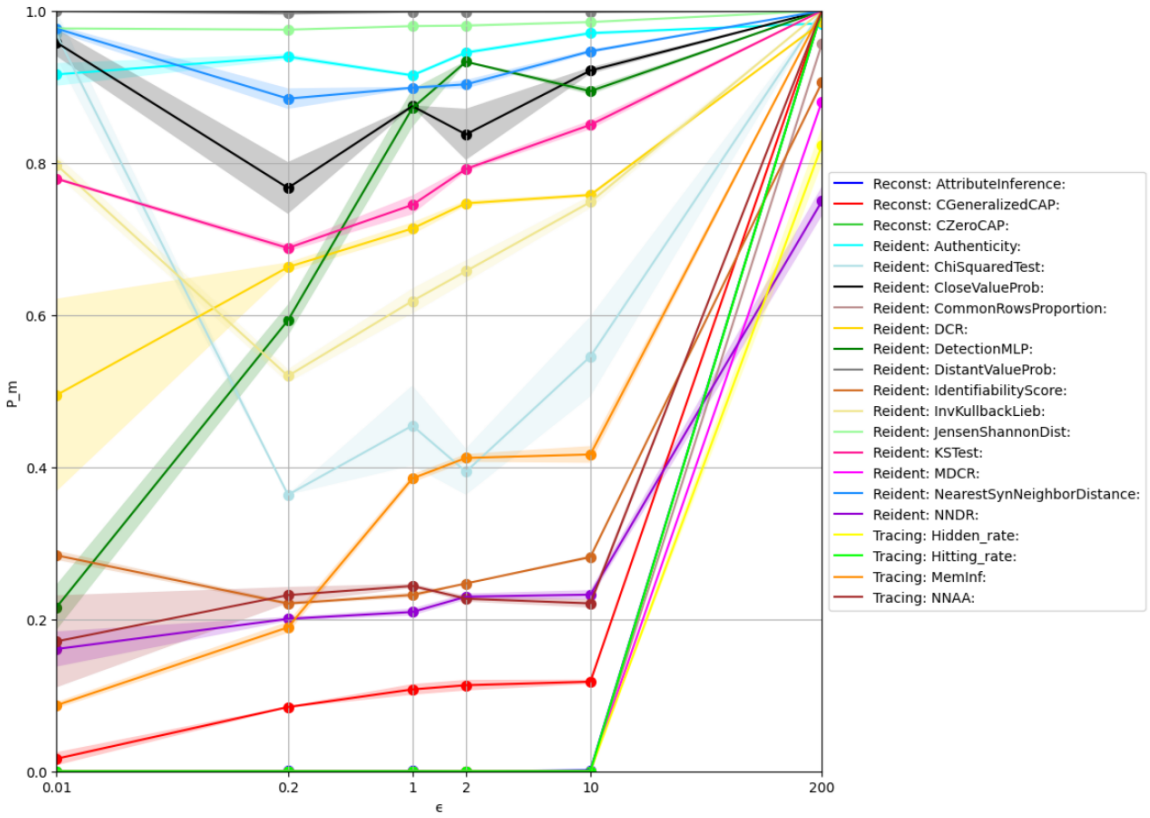

Fig. 9. Metric results for PrivBayes on MedOnc with different values of epsilon, with logarithmic ϵ axis. The endpoints for ϵ, however, are the Random and Real results, with ϵ set to 0.01 and 200, respectively. The exact choice of these endpoint ϵ values was mostly for visual purposes; theoretical values are discussed in Section 6.3. The deviation is SEM, and is shown with the translucent areas around each line

The results for MedOnc with PrivBayes in Figure 9 show that most of the metrics lie in between the ends (Random and Real) except for CloseValueProb, NearestSyntheticNeighbourDistance, ChiSquaredTest, IdentifiabilityScore and InvKullbackLieb.

DetectionMLP and Membership Inference seems to capture the expected tendency the most, i.e. being between the ends and having a big change in the middle of the graph.

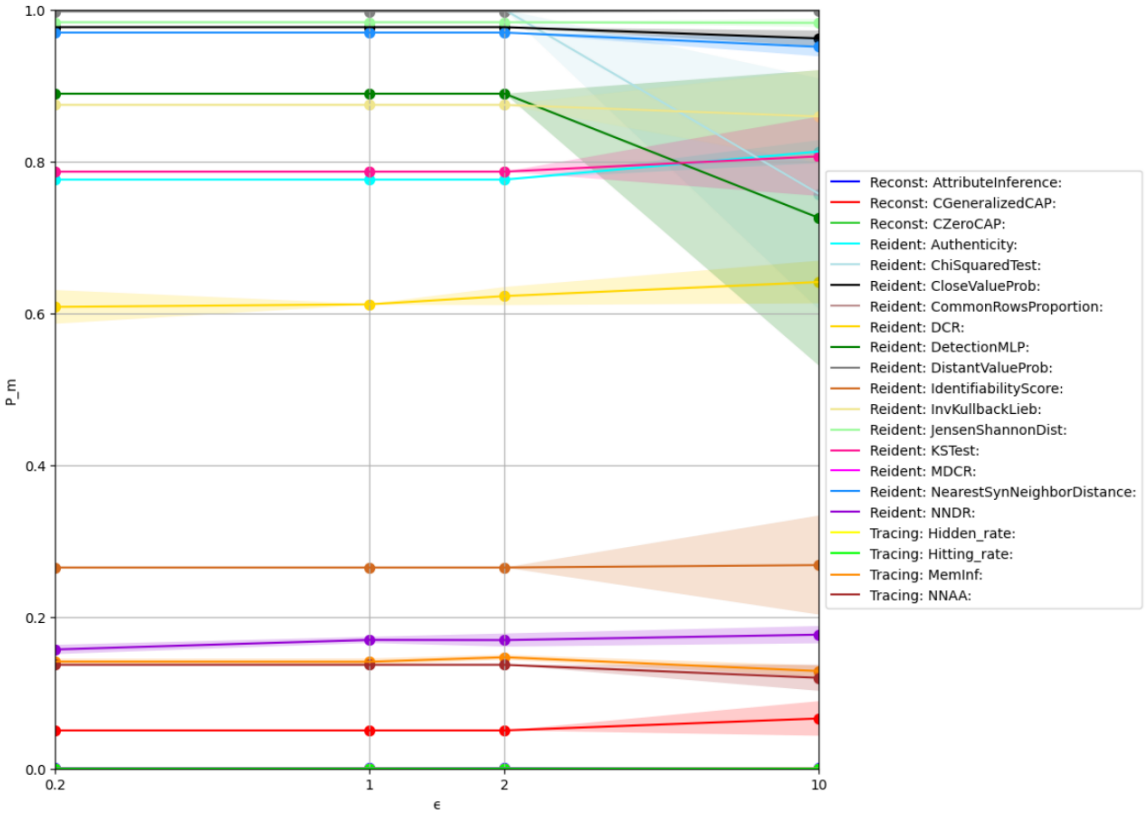

Fig. 10. Metric results for PATE-GAN on MedOnc with different values of epsilon, with logarithmic ϵ axis. The deviation is SEM, and is shown with the translucent areas around each line

For PATE-GAN in Figure 10, there is not much variance between the individual metric scores for the different ϵ configurations. For ChiSquaredTest and DetectionMLP from $\epsilon = 2$ to $\epsilon = 10$ there is a significant drop in the mean as well as extremely high deviation for both. With the rest of the metrics only slight deviation can be seen.

To summarise, the MedOnc dataset results are mostly the same as the Texas dataset, with the hypothesis mostly holding true for PrivBayes, but not with PATE-GAN, which displays no significant change between the individual ϵ configurations.

This means that for Experiment 1.2, PrivBayes confirmed the hypothesis, which showcased a clear monotonic increase for most of the metrics. PATE-GAN did not confirm it, where most metrics showed no change across the different ϵ configurations. This we believe to be an implementation error. Excluding the results showcased with PATE-GAN, we deem the following metrics to work as showcased by Table 8. This was determined based on both Figure 7 and Figure 9.

Works adequately	**Sometimes works**	**Inadequate functionality**
• CGeneralizedCAP	• Hitting Rate	• Attribute Inference
• DetectionMLP	• Authenticity	• CZeroCAP
• MemInf	• InvKullbackLieb	• IdentifiabilityScore
• DCR	• KSTest	• JensenShannonDist
	• MDCR	• NNDR
		• Hidden Rate
		• NNAA
		• ChiSquaredTest
		• CloseValueProb
		• CommonRowsProportion
		• DistantValueProb
		• NearestSynNeighborDistance

Table 8. Metric functionality breakdown for Experiment 1.2

7.3 Experiment 2.1: Spearman

The hypothesis for Experiment 2.1 is that metrics within the same attack category will have a stronger correlation than those of different categories, indicating that the categorisation is meaningful in terms of the outputs of the metrics.

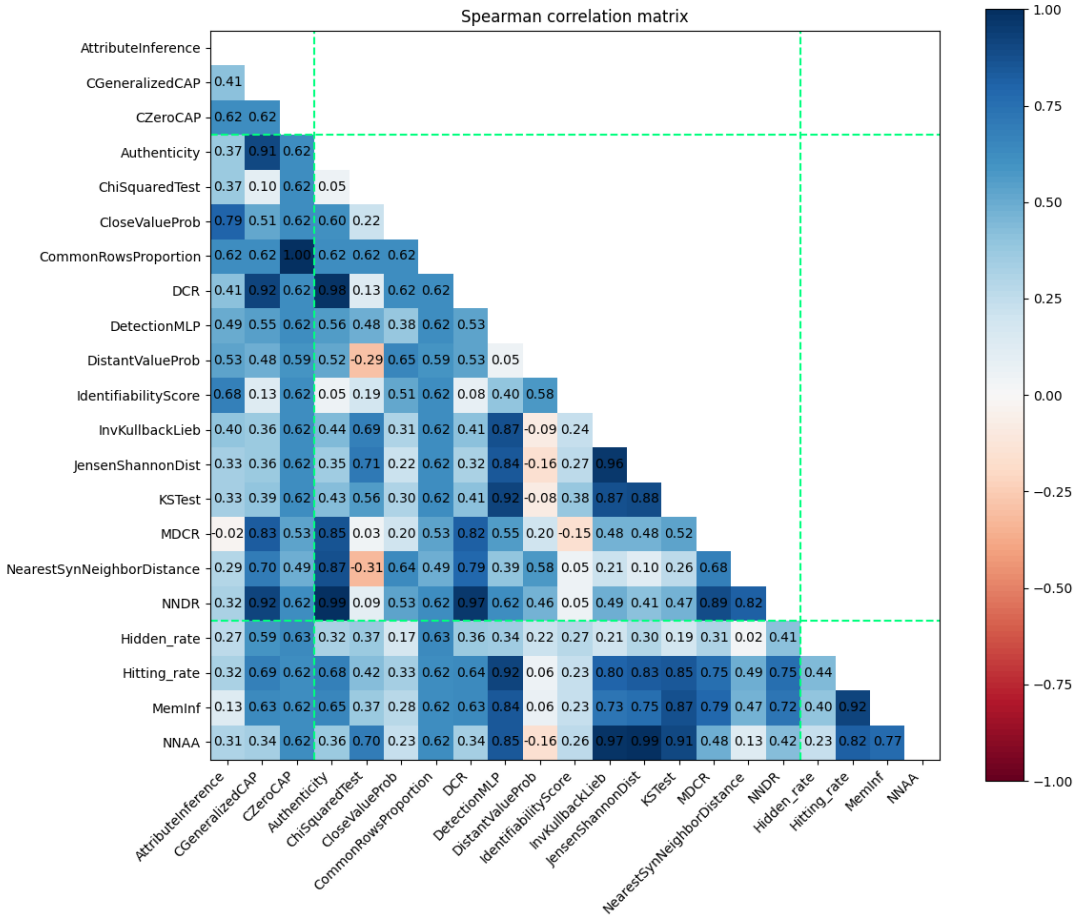

Fig. 11. Correlation matrix for the metrics run on the Texas dataset, using Spearman correlation. Each square is coloured according to its own value, and on each axis the attack category boundaries are shown with green lines

In Figure 11, the correlation matrix for metrics using the Texas data is shown. Almost all metric pairs have positive correlations, which is expected. However, there seems to be no clear difference in (1) the correlation strength within an attack category and (2) the correlation strength between different attack categories. We expected to see that the top-left, middle and bottom-right areas (as outlined by the green lines) would have higher correlation than the other areas, but if this is present it is not nearly as clear as expected.

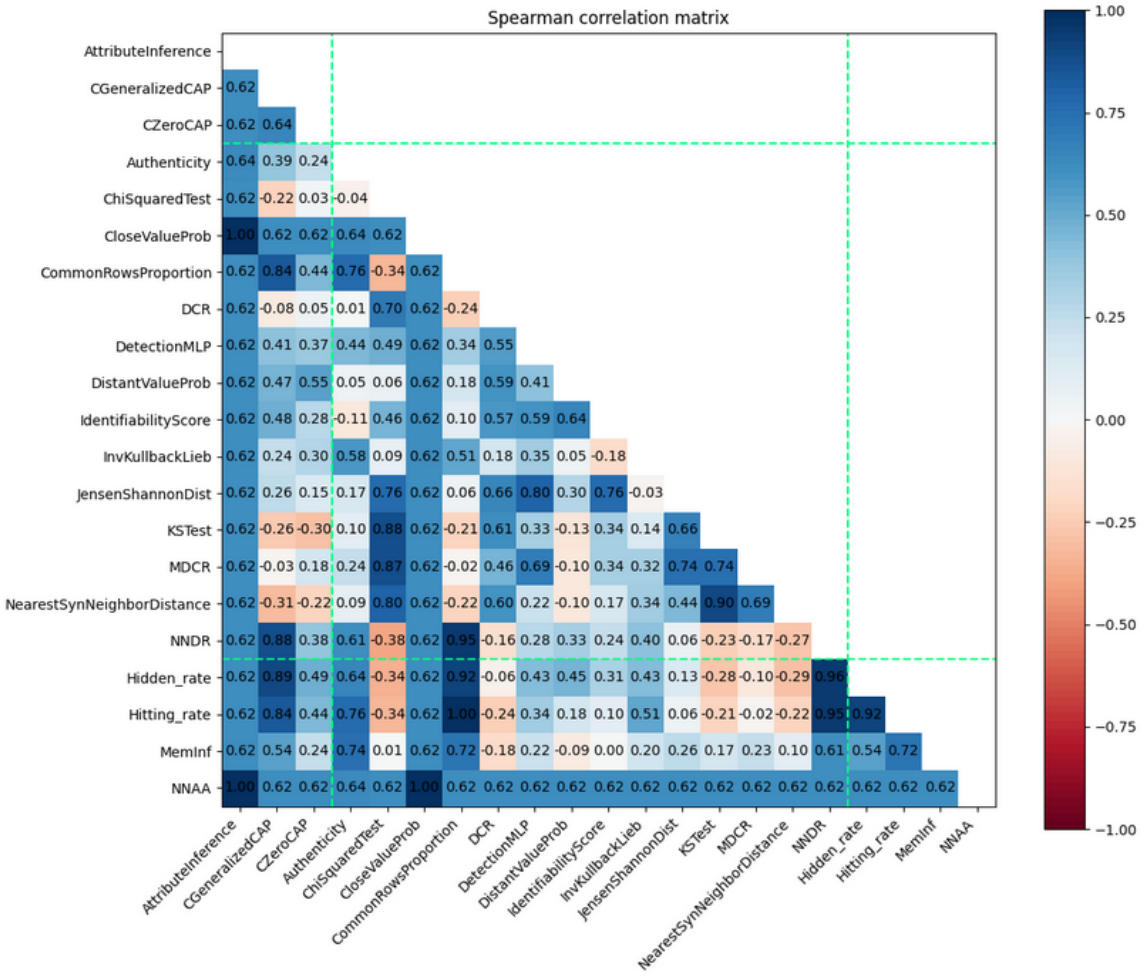

Fig. 12. Correlation matrix for the metrics run on the MedOnc dataset, using Spearman correlation. Each square is coloured according to its own value, and on each axis the attack category boundaries are shown with green lines

For MedOnc, the correlation matrix for the metrics is shown in Figure 12. Once again most of the metrics have a positive correlation, as expected, though the attack category boundaries do not seem to be reflected in the correlations.

To summarise Experiment 2.1, the hypothesis here seems to be disconfirmed. This means that the categorisation of the metrics is not reflected in their results, seeing as the correlation does not seem to differ between (1) metrics within the same category and (2) metrics from different categories.

7.4 Experiment 2.2: Clustering

The hypothesis for this experiment is that the metrics will cluster according to the attack categories considered, and thereby form three clusters. In any case, clustering of metrics could help to further indicate which metrics behave similarly, and thereby which ones are unnecessary to test. Similar to Experiment 2.1, this would indicate that the categorisation of the metrics is reflected in the metrics' results.

Outliers:

- ChiSquaredTest

Cluster 0:

- AttributeInference
- DetectionMLP
- NNDR
- Hitting_rate
- MemInf
- NNAA

Cluster 1:

- CGeneralizedCAP
- Authenticity

Cluster 2:

- CZeroCAP
- CommonRowsProportion
- Hidden_rate

Cluster 3:

- CloseValueProb
- DCR
- DistantValueProb
- IdentifiabilityScore
- InvKullbackLieb
- JensenShannonDist
- KSTest
- MDCR
- NearestSynNeighborDistance

Table 9. Clustering results for the Texas dataset

The clustering results for the Texas dataset are shown in Table 9. To reiterate, this clustering is based on the full results, i.e. what is shown in Figure 5. However, that figure shows colours based on the ranking of each method, for each metric. In order to better visualise how these clusterings make sense, in Appendix A.2 this figure of full results is shown again, but where the colours are based directly on the value shown. For Texas, this is Figure 13.

The way OPTICS works is essentially by clustering based on the density of points. As an example, from Figure 13 we can see that the results for CZeroCAP, CommonRowsProportion and Hidden rate are close to equal. This means that those three points have high density, and they are therefore clustered together in cluster 2. Because the density is so high, CGeneralizedCAP and Authenticity are not included in this cluster, despite their results also being close. The other clusters are not as obvious, and are therefore not discussed further in-depth.

Aside from outliers, four clusters are identified by OPTICS. This does not quite match the hypothesis, but is close. It is however unexpected that there would be an outlier, specifically ChiSquaredTest. From Figure 13 it is not clear why this was calculated to be an outlier - perhaps another form of visualisation could help, though it may simply be difficult due to the somewhat high dimensionality (12D) of the data being clustered.

Outliers:

- ChiSquaredTest
- DetectionMLP
- AttributeInference

Cluster 0:

- Hitting_rate
- CommonRowsProportion
- CZeroCAP

Cluster 1:

- Hidden_rate
- MDCR

Cluster 2:

- MemInf
- NNAA
- IdentifiabilityScore
- NNDR
- CGeneralizedCAP

Cluster 3:

- KSTest
- Authenticity
- InvKullbackLieb
- DCR

Cluster 4:

- CloseValueProb
- NearestSynNeighborDistance

Cluster 5:

- DistantValueProb
- JensenShannonDist

Table 10. Clustering results for the MedOnc dataset

In Figure 10 the clustering results for MedOnc can be seen. Here, six clusters in total were located with ChiSquaredTest, DetectionMLP and AttributeInference as outliers.

Some of the clusters seem well grouped. This includes cluster 0, where it can be observed from Figure 14 that the metric scores are very similar. In cluster 1 however, Hidden_rate and MDCR seem quite distant. The remaining clusters are not discussed as they are not easily interpretable due to the somewhat high data dimensionality again.

From there being six clusters and three outliers, the clustering for MedOnc is not as indicative of the hypothesis being true.

In summary, the hypothesis for Experiment 2.2 can not be confirmed in respect to the metrics being clustered according to their attack category. However, the produced clusters ended up being quite similar between the two datasets. Here, the same metrics were mostly clustered between the datasets, which e.g. includes CZeroCAP and CommonRowsProportion.

7.5 Experimental summary

Now that all results have been showcased, we want to be able to conclude, which metrics are adequate to evaluate privacy of anonymised data, based on our findings. We determine this by looking at the results of each experiment and then eliminating individual metrics.

We start by removing the metrics that show inadequate functionality in either of the first set of experiments, which are displayed as Inadequate functionality in Tables 7 and 8. The remaining metrics after this are in Table 11.

Reconstruction metrics

- CGeneralizedCAP

Re-identification metrics

- DetectionMLP
- DCR
- Authenticity
- InvKullbackLieb
- KSTest
- MDCR

Tracing metrics

- MemInf
- Hitting Rate

Table 11. List of metrics after 1st round of exlusions

Reconstruction metrics

- CGeneralizedCAP

Re-identification metrics

- DetectionMLP
- DCR
- Authenticity
- MDCR

Tracing metrics

- MemInf
- Hitting Rate

Table 12. List of metrics after 2nd round of exlusions

For choosing between these, the Spearman correlations do not seem to give much insight, as the pairs of metrics with high correlation have no overlap between Figures 11 and 12. Using the clustering, however, some metrics are clustered together for both of the datasets. From cluster 2 in MedOnc, we could select between MemInf or CGeneralizedCAP, but seeing as both of these are categorised as working well in the first set of experiments, we deem that both should be kept. From both Texas cluster 3 and MedOnc cluster 3, DCR, InvKullbackLieb and KSTest are present. Out of these, only DCR is deemed to work well, and as such we can exclude InvKullbackLieb and KSTest. Then from cluster 1 in Texas, Authenticity could be excluded as it it deemed less functional than CGeneralizedCAP, though for MedOnc they are not clustered together and it is therefore kept, with a few other similar cases being present. This leads to the list in Table 12 containing the metrics the results deemed to be necessary to evaluate privacy of an anonymised dataset. From this list we can see that all attack categories are represented.

8 DISCUSSION AND FUTURE WORKS

8.1 Results

For the first set of experiments (Experiments 1.1 and 1.2), the results showed some variation between the datasets where some metrics performed as expected in one dataset, but not in the other. Ultimately, for both Experiments 1.1 and 1.2, four metrics were found to work adequately (though not the exact same). For Experiment 1.2, the results for varying ϵ using PrivBayes show promise, especially with the Texas dataset. Even this result is far from ideal though, as we would expect that the largest difference happens around an ϵ value of 0.2 to 10, whereas most metrics show little difference within this range. Part of this likely comes down to our tuning of the metrics. This could have been improved in several ways, such as implementing fewer metrics and spending more time tuning each metric. As opposed to PrivBayes, PATE-GAN showed no significant change between the different ϵ configurations, which could be caused by an implementation error. This means we only had PrivBayes as a functioning DP synthesiser. We had also intended to include more methods, alongside other limitations regarding anonymisation techniques.

For the second set of experiments (Experiments 2.1 and 2.2), the results were not as expected. With the Spearman correlation we did not gain much insight as the results did not indicate any clear pattern of correlation between the metrics. Generally, the metrics were correlated, but not in a discernible pattern and ultimately did not help much in the final selection of metrics in Section 7.5. However, with clustering, we were able to further narrow down the metrics where some similarity was showcased between the clustering for Texas and MedOnc. It could perhaps have been better to cluster based on the rankings of the metric scores, rather than the raw metric outputs. In this case the clusterings would moreso reflect the Spearman correlations shown in Experiment 2.1. For Texas e.g. DistantValueProb and IdentifiabilityScore are considered very distant (see Figure 13), despite them having a Spearman correlation of 0.64 which is fairly high (see Figure 11 and 5).

Using the results of the two experiments, we were able to narrow down the list of metrics from 21 to 7. These final 7 covers all attack categories ensuring a comprehensive evaluation of privacy risks. The list could be further narrowed down, including Authenticity and Hitting Rate, but we ultimately decided not to do so as the clusterings of the two datasets were contradicting. Therefore, it would be good to have a third dataset cluster analysis which we could use in the metric selection process to help further rule out metrics in such instances. Another idea, that could have helped the metric selection process, could be to exclude inadequate metrics already after Experiment 1.1. This could help simplify the later experiments, improve accuracy of the results, as well as making the following figures easier to interpret.

The problem statement defined in Section 1 was answered, where we were able to extract a list of privacy metrics which cover the relevant privacy attacks. These metrics are able to capture the privacy level of a given synthesiser, regardless of the synthesiser having privacy mechanisms or not.

8.2 Scope

In Section 3, we intended to include a term for additional inputs to the metrics, such that the definition of the privacy metrics could cover the need for additional input requirements. This could for example be the set of attributes A, but should also cover e.g. being able to input the method used to generate the synthetic data. This would be a requirement for some metrics such as those by Stadler et. al. [16].

The algorithms in Section 5 could also be more polished, though further improvement to this was infeasible within the scope due to the number of metrics and many of them being complex. If we had known this earlier in the making of

this paper, we would likely have opted to include fewer metrics in favor of improving the quality of the algorithms and descriptions.

We had also planned to include the MIMIC dataset [39], but decided to remove it from the experiments, to get more runs and thereby more accuracy with the other two datasets, at the cost of generalisability.

For the MedOnc dataset, there may have been better ways to sample it. As mentioned in Section 6.1, we used all tuples without NA values, though this was only a small portion of the full dataset. Instead one could e.g. replace NA values with other valid values (Imputation), and then use a sampling technique to choose a subset of the patients, and using all the tuples associated with those patients. This could yield a larger dataset used in the experiments, which may improve the accuracy of the results. Furthermore, it could be that there is a bias as to which rows contain NA values, which this technique could reduce or eliminate, and is therefore a good contender for future work.

For the Spearman correlation experiment, we could also have calculated a form of variation, such as confidence intervals. This could for example be by using the Jackknife Euclidean likelihood approach [40]. This could help to identify how certain we are in the correlation values and thereby whether it would make sense to perform more runs to increase accuracy for this.

8.3 Anonymisation techniques

We had initially planned to include more anonymisation techniques to get more generality, but excluded some partly due to difficulties with the code, and partly due to computation limitations, therefore favoring more runs with fewer methods. For methods with a privacy mechanism, we had intended to include DPGAN [41] and ADS-GAN [42], but in our code both of these had issues similar to those seen with PATE-GAN in Figure 8 where the privacy-adjusting mechanism seems to have little to no impact on the results. For methods without an explicit privacy mechanism, we intended to include PAR [24] and Bayesian_network [9], but both of these also had implementation/adjustment issues. For PAR the problem is that it specifically generates time series data where some kind of sequence id is required (e.g. patient id). For Bayesian_network it simply produced invalid data, which the metrics could not accept as input.

Our Random baseline could have been simpler and performed better. The best option for this may have been to ask an expert who is independent of the dataset about the possible set of values for each attribute, and then randomly generate values within those sets of values. For example for dates, we could ask which range of dates is possible, and generate randomly within that range. However, this was discovered fairly late in the process, at which point we had already run a large part of the experimental data for the Random baseline, and it was also uncertain what time would be required to get this information. The reason for the complexity of the algorithm used is in part due to it being a further development of the Random baseline used in our prior work [7]. It seems the results with MedOnc may have been negatively impacted by this, as in Figure 6 Random does not seem to perform the best, with notably PrivBayes ($\epsilon = 0.2$) often performing better. This could however also be in part due to inadequate metric tuning as discussed in Section 8.4.

For all of the methods used, they could likely have been better tuned for the datasets. This could for example include changing the exact details of networks for GANs, but this was a low priority as this paper focuses more on the metrics.

8.4 Metrics

For this paper one of our goals was to test with more metrics than in our prior work [7], as metrics are once again the focus. This goal has partially been met, though most of the additions have been to the reidentification category which already had the largest number of metrics. This is because it has been difficult to find more privacy metrics in the other two categories, and it has been difficult to get them working well enough to include them. For example, we attempted

to include the "Linkability privacy game" and "Attribute inference privacy game" by Stadler et. al. [16], but transferring their metric code to our codebase turned out to be too difficult due to rather different codebase structures, and likewise for transferring our anonymisation technique code, datasets and more into their codebase.

We have also categorised the metrics according to the privacy attack we deem they most closely measure the risk of, based on how the metric works. During the process of writing we gained more insight into how the metrics work, with which we discovered, for example, that Hidden Rate seems to have been categorised wrongly. It was categorised as a Tracing metric (i.e. checking if each sensitive tuple is included in the anonymised dataset), but it functions by checking for each sensitive tuple whether that is closest to its (assumed) associated anonymised tuple, which should have been categorised as a Reidentification metric. The reason for not changing it upon discovering the error is due to it being discovered late in the experimentation. Furthermore, categorising the metrics has often proven difficult, as they may not clearly fall into one category or another. This is also part of the reason for introducing the clustering experiment, where we make no prior assumption about which category each metric is associated with.

As with the methods, some of the metrics also require tuning to work properly and again the metrics could likely be tuned better. In this case it has been a balancing act between tuning the metrics better for accuracy and adding more metrics for generality. For example, in Figure 7 there is great discrepancy in the ϵ value around which the metric changes most, where many of the metrics have the greatest change at one of the extreme ends of the graph. For this tuning we could also have used an automated approach, similar to hyperparameter tuning, which could be material for future works.

9 CONCLUSION

In conclusion, we set out to investigate the state-of-the-art of privacy metrics used to evaluate the privacy of anonymised data. We found that the current literature does not have a common privacy evaluation methodology which makes it hard to compare the results. Additionally, not much attention is paid to privacy evaluation as most of the focus is on utility. Therefore, our goal was to investigate which privacy metrics are needed in order to adequately evaluate privacy of anonymised data. To study this, we performed two sets of experiments; The first set, which examined whether the metrics works as intended, i.e. whether the metric would output a score that is relative to the privacy level of a given anonymised data and the seconds set, which tested whether different metrics capture different aspects of anonymisation of a given dataset. Here, the second set of experiments was used to see if some metrics were behaving the same in terms of their scores and if so, whether we could exclude them, further narrowing down the list of metrics. Spearman correlation and clustering analyses were used for this purpose.

In the end, we were able to provide a list of seven metrics which we deemed to be sufficient for evaluating anonymised data, based on our results. This list covers all privacy attacks relevant to our scope. Further work can be conducted to generalise the results such as introducing a third dataset to the experiments.

ACKNOWLEDGMENTS

We would like to express our gratitude to our supervisors, Daniele Dell'Aglio, Martin Bøgsted, and Jakob Bruhn Krøjgaard Skelmose for their guidance, insight, expertise, and support throughout the duration of this project. Their encouragement and patience helped us navigate through the project. Furthermore, we would like to thank Aalborg University Hospital for getting access to the MedOnc dataset as well as Martin Aumüller and Nadia Krag for assisting in the implementation of some privacy metrics.

REFERENCES

[1] Shaden Al-Aqeeli, Mznah Al-Rodhaan, and Yuan Tian. Privacy preserving risk mitigation strategy for access control in e-healthcare systems. In *2017 International Conference on Informatics, Health & Technology (ICIHT)*, pages 1–6, February 2017.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[3] Charu Aggarwal. On k-anonymity and the curse of dimensionality. volume 2, pages 901–909, 01 2005.

[4] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! A Survey of Attacks on Private Data. *Annual Review of Statistics and Its Application (2017)*, 2017.

[5] Hajra Murtaza, Musharif Ahmed, Naurin Farooq Khan, Ghulam Murtaza, Saad Zafar, and Ambreen Bano. Synthetic data generation: State of the art in health care domain. *Computer Science Review*, 48:100546, 2023.

[6] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493, 04 2022.

[7] Anders Martin Hansen, Frederik Stær, Frederik Marinus Trudslev, and Silas Oliver Torup Bachmann. An experimental comparison of privacy metrics. https://kbdk-aub.primo.exlibrisgroup.com/permalink/45KBDK_AUB/a7me0f/alma9921650768405762, 2024.

[8] Chao Yan, Yao Yan, Zhiyu Wan, Ziqi Zhang, Larsson Omberg, Justin Guinney, Sean D. Mooney, and Bradley A. Malin. A multifaceted benchmarking of synthetic electronic health record generation models. *Nature Communications*, 13(1), dec 2022.

[9] Zhaozhi Qian, Rob Davis, and Mihaela van der Schaar. Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[10] Judith Sáinz-Pardo Díaz and Álvaro López García. A python library to check the level of anonymity of a dataset. *Scientific Data*, 9(1):785, Dec 2022.

[11] Tobias Hyrup, Anton Danholt Lautrup, Arthur Zimek, and Peter Schneider-Kamp. Sharing is cairing: Characterizing principles and assessing properties of universal privacy evaluation for synthetic tabular data, 2023.

[12] Anton Danholt Lautrup, Tobias Hyrup, Arthur Zimek, and Peter Schneider-Kamp. Syntheval: A framework for detailed utility and privacy evaluation of tabular synthetic data, 2024.

[13] ARX. Arx - data anonymization tool.

[14] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. Database System Concepts - 7th edition — db-book.com. https://db-book.com/, 2019.

[15] Damien Desfontaines. A friendly, non-technical introduction to differential privacy. https://desfontain.es/privacy/friendly-intro-to-differential-privacy.html, 09 2021. Ted is writing things (personal blog).

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

[17] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. In *Foundations and Trends in Theoretical Computer Science*, 2014.

[18] LATANYA SWEENEY. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[19] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018.

[20] Maria Rigaki and Sebastian Garcia. A Survey of Privacy Attacks in Machine Learning. *ACM Computing Surveys*, 56(4):1–34, April 2024. arXiv:2007.07646 [cs].

[21] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, page 202–210, New York, NY, USA, 2003. Association for Computing Machinery.

[22] Jinsung Yoon, Michel Mizrahi, Nahid Farhady Ghalaty, Thomas Jarvinen, Ashwin S. Ravi, Peter Brune, Fanyu Kong, Dave Anderson, George Lee, Arie Meir, Farhana Bandukwala, Elli Kanal, Sercan Ö. Arık, and Tomas Pfister. Ehr-safe: generating high-fidelity and privacy-preserving synthetic electronic health records. *npj Digital Medicine*, 6(1), August 2023.

[23] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

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

[25] Jinsung Yoon, Lydia N. Drumright, and Mihaela van der Schaar. Anonymization Through Data Synthesis Using Generative Adversarial Networks (ADS-GAN). *IEEE journal of biomedical and health informatics*, 24(8):2378–2388, August 2020.

[26] Morgan Guillaudeux, Olivia Rousseau, Julien Petot, Zineb Bennis, Charles-Axel Dein, Thomas Goronflot, Nicolas Vince, Sophie Limou, Matilde Karakachoff, Matthieu Wargny, and Pierre-Antoine Gourraud. Patient-centric synthetic data generation, no reason to risk re-identification in biomedical data analysis. *npj Digital Medicine*, 6(1):37, Mar 2023.

[27] David S Moore, George P McCabe, and Bruce A Craig. *Introduction to the Practice of Statistics*. W.H. Freeman, 2012.

[28] Frank Nielsen. On the jensen-shannon symmetrization of distances relying on abstract means. *Entropy*, 21(5):485, 2019.

[29] GitHub - octopize/saiph: A projection package — github.com. https://github.com/octopize/saiph. [Accessed 11-06-2024].

[30] Lars Børty, Rasmus F. Brøndum, Heidi S. Christensen, Charles Vesteghem, Marianne Severinsen, Søren P. Johnsen, Lars H. Ehlers, Ursula Falkmer, Laurids Ø. Poulsen, and Martin Bøgsted. Trends and drivers of pharmaceutical expenditures from systemic anti-cancer therapy. *The European Journal of Health Economics*, 24(6):853–865, Aug 2023.

[31] Public Use Data File (PUDF) Inpatient Free Download | Texas DSHS — dshs.texas.gov. https://www.dshs.texas.gov/texas-health-care-information-collection/general-public-information/hospital-discharge-data-public/Public-Use-Data-File-(PUDF)-Inpatient-Free-Download. [Accessed 12-03-2024].

[32] Douglas G Altman and J Martin Bland. Standard deviations and standard errors. *BMJ*, 331(7521):903, 2005.

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

[34] Cuong Tran, My H. Dinh, Kyle Beiter, and Ferdinando Fioretto. A fairness analysis on private aggregation of teacher ensembles, 2021.

[35] 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.

[36] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan, 2019.

[37] Royal Geographic Society. A guide to spearman's rank.

[38] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, SIGMOD '99, page 49–60, New York, NY, USA, 1999. Association for Computing Machinery.

[39] Alistair Johnson, Lucas Bulgarelli, Tom Pollard, Steven Horng, Leo Anthony Celi, and Roger Mark. Mimic-iv, 2023.

[40] Miguel de Carvalho and Filipe J. Marques. Jackknife euclidean likelihood-based inference for spearman's rho. *North American Actuarial Journal*, 16(4):487–492, 2012.

[41] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network, 2018.

[42] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, jun 2018.

A APPENDIX

A.1 Dataset preprocessing

For all the datasets, adjustments may be made to whether they are comma- or semicolon-delimited.

We acquired MedOnc from the Aalborg University hospital Oncology Department through the UCloud platform, with the help of our supervisors. The preprocessing of MedOnc consists of:

- Converting the date format to Unix time
- Adjusting decimal values to have a "." instead of ","
- Removing rows containing no values or "NA" values

We acquired the Texas dataset from the public Github repository: https://github.com/spring-epfl/synthetic_data_release. This was accessed 12-03-2024. For Texas, we did not do any preprocessing.

A.2 Full results, linear instead of ranking based

In Section 7.1 the full results are shown, with colours based on the ranking of the methods, for each metric. In this appendix, they are shown again but with the colours based on the individual metric values, to aid in comparison between metrics.

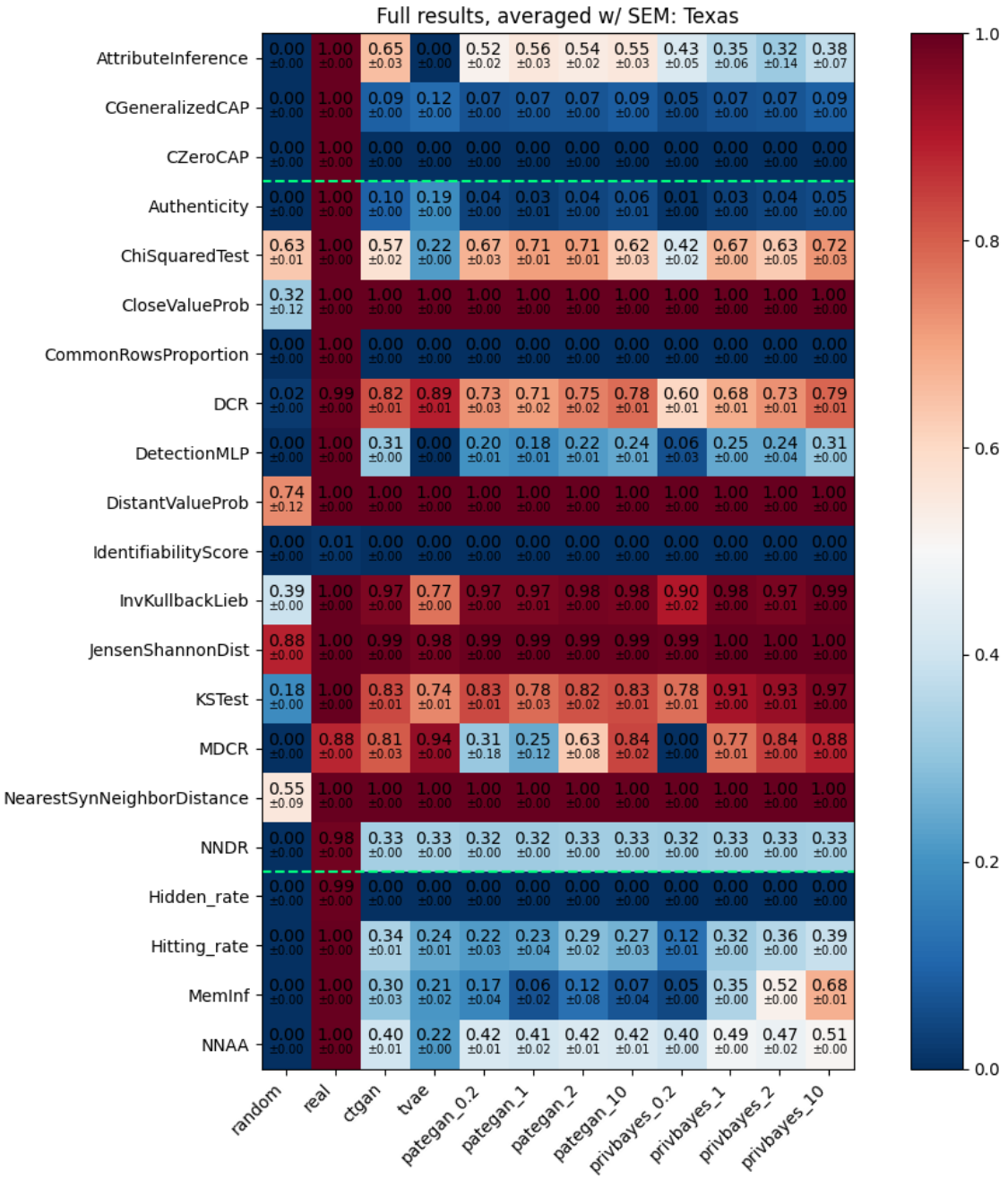

Fig. 13. Privacy metric scores on anonymised datasets generated from the Texas dataset. In each cell, the value at the top is the average of the runs, while the value below is the deviance, specifically SEM. The value in each cell determines the colour of for that cell, with the highest score (1) being red, and lowest score (0) being blue. The green lines indicate the boundaries between different attack categories

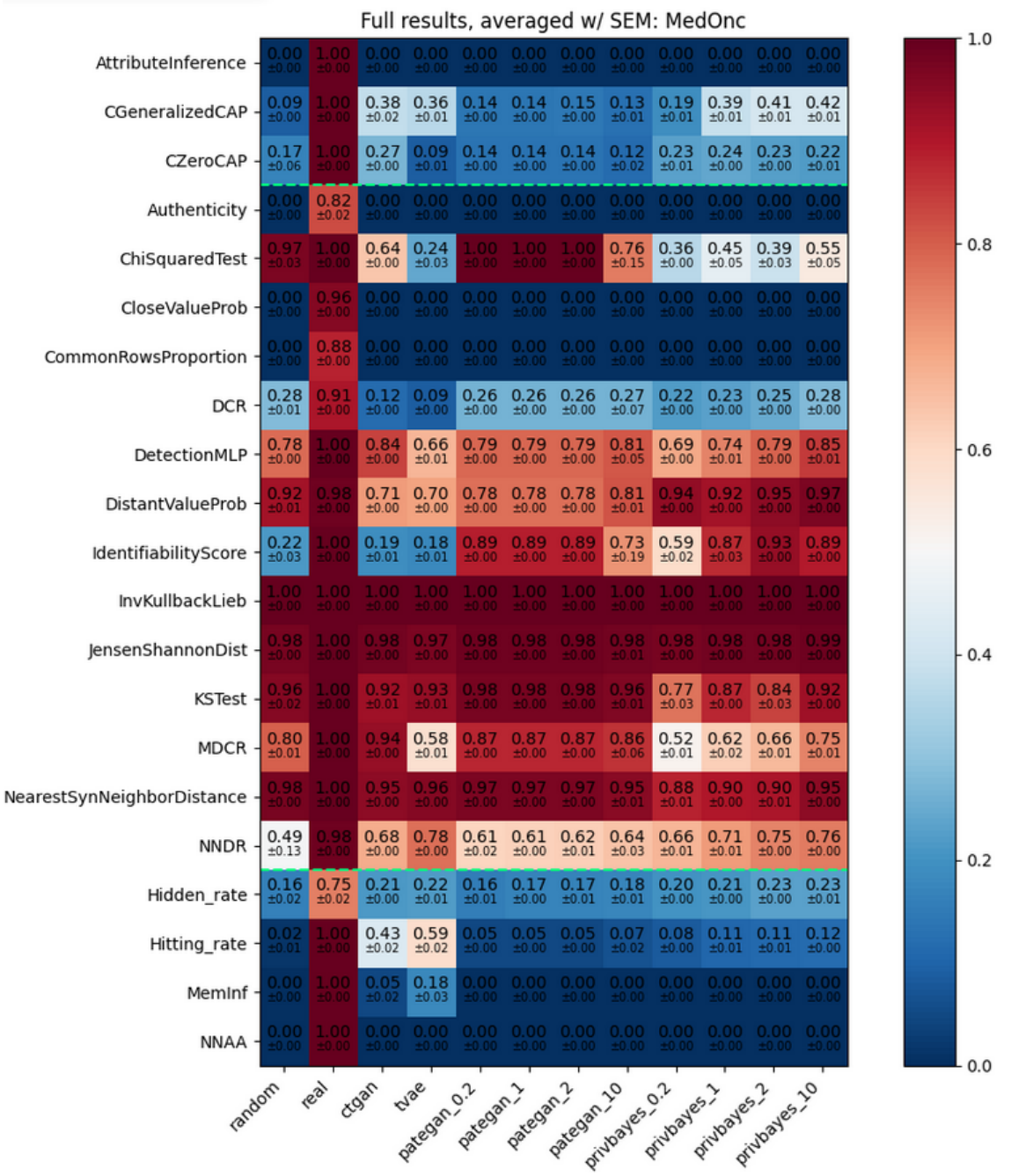

Fig. 14. Privacy metric scores on anonymised datasets generated from the MedOnc dataset. In each cell, the value at the top is the average of the runs, while the value below is the deviance, specifically SEM. The value in each cell determines the colour of for that cell, with the highest score (1) being red, and lowest score (0) being blue. The green lines indicate the boundaries between different attack categories