Efficient Bayesian Network Construction for Increased Privacy on Synthetic Data

Markus Hittmeir
SBA Research, Vienna, Austria
mhittmeir@sba-research.org

Rudolf Mayer
SBA Research, Vienna, Austria
rmayer@sba-research.org

Andreas Ekelhart
SBA Research, Vienna, Austria
aekelhart@sba-research.org

***Abstract*—The use of synthetic data is a widely acknowledged privacy-preserving measure that reduces identity and attribute disclosure risks in micro-data. The idea is to learn the statistical properties of an original dataset, store this information in a model, and then use this model to generate artificial samples and build a synthetic dataset that resembles the original. One of the many different approaches of synthetization tools relies on describing the original dataset by using a Bayesian network. This method is implemented in the open-source tool *DataSynthesizer* and has proven particularly suitable for datasets with a small to moderate number of attributes. In this paper, we will substitute the greedy algorithm used for learning the Bayesian network by a substantially faster genetic algorithm. In addition, our goal is to protect particularly sensitive attributes by decreasing specific correlations in the synthetic data that may reveal personal information. We will thus show how to customize the network structures for specific machine learning tasks. Our experiments demonstrate that this technique allows to further decrease the disclosure risks and, hence, add to the applicability of synthetic data as technique for privacy preservation.**

***Index Terms*—Synthetic Data, Machine Learning, Genetic Algorithms, Privacy, Disclosure Control**

I. Introduction

Micro-data, i.e. data that contains information about individuals, is collected in domains such as health care, employment, or social media. Advances in data analysis and an increasing interest in accessing and mining micro-data constitute a threat to the privacy of the affected individuals. Several regulations, such as the EU's General Directive on Data Protection (GDPR), put restrictions on how data can be collected and shared. Traditional approaches to comply with data protection and privacy aspects often include anonymization of data before publishing or processing, such as in the approaches of *k-anonymity* [1] or *differential privacy* [2].

An increasingly well-studied addition to traditional privacy-preserving techniques is the generation of *synthetic data*. Here, the idea is to learn the global properties of an original dataset (that cannot easily be shared), and generate a synthetic dataset thereof. The aim is to preserve the distributions and correlations between the attributes while simultaneously avoiding to reveal the individuals described by the data. The synthetic data can then be shared with considerably reduced privacy risks and holds the promise to allow data analysis with similar results. One of the earliest usages of synthetic data was in the partial synthetic data approach by Rubin [3], where certain columns are generated synthetically. An overview on more than 20 fully and partially synthetic data approaches is given in [4].

In this paper, we will focus on *fully* synthetic data, meaning that all columns of the original dataset are synthesized. Nowadays, there is a large variety of commercial and open-source tools (e.g., [5], [6], [7]). The different approaches include fitting Gaussian Copulas or training Generative Adversarial Networks (GANs). Another possibility is the construction of Bayesian networks, which is used in the *DataSynthesizer* (DS) ([5])[1]. This approach is based on the earlier work *PrivBayes* [8] for learning the correlations between attributes on the original dataset. A greedy algorithm is employed to construct a Bayesian network as part of the description of the global properties of the original dataset. While the approach has shown a high suitability for small to medium-sized datasets, the runtime complexity of the greedy algorithm increases drastically with the dimensions of the dataset. The **first contribution** of this paper is thus an efficient implementation suited for Big Data, with the substitution of the greedy algorithm by a genetic algorithm[2]. Our experiments in Section V-A show that our adaptation yields a substantial improvement in efficiency.

In the case of micro-data, two risks of disclosure of sensitive information are widely considered. *Identity disclosure* happens when an adversary is able to conclude that a certain record in the dataset belongs to a certain individual. Due to the nature of fully synthetic data generation approaches, there is no 1-to-1 link between real individuals and synthetic data samples. As a result, identity disclosure in the defined sense is not possible. Previous publications about privacy risks on fully synthetic data ([9], [10]) have thus focused on the notion of *attribute disclosure*, which refers to the risk that, on structured tabular datasets, an adversary might learn the victim's value of a sensitive attribute. It has been concluded that attribute disclosure can happen without identity disclosure, and that there remains a risk for attribute disclosure on synthetic data. While the original implementation of the DataSynthesizer allows to construct differentially private synthetic data, previous

This work was partially funded by the Austrian Research Promotion Agency FFG under grant 877173 (GASTRIC). SBA Research (SBA-K1) is a COMET Centre within the framework of COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna. The COMET Programme is managed by FFG.
978-1-6654-8045-1/22/$31.00 ©2022 IEEE

[1]Source code: https://github.com/DataResponsibly/DataSynthesizer
[2]Implementation: https://github.com/sbaresearch/EnhancedDataSynthesizer

evaluations ([11], [12], [13]) have shown a noticeable drop in the utility when differential privacy is enabled compared to the performance when it is disabled. The **second contribution** of this paper is thus a novel approach for decreasing disclosure risk, namely customized Bayesian networks. The main idea is to conceal particular correlations in the synthetic data to prevent the adversary from learning reliable information about sensitive attributes. Our experiments demonstrate that this can be achieved without a substantial loss in utility of the data for machine learning. As a result, customized Bayesian networks may be a suitable addition to the toolkit of measures for statistical disclosure control.

The remainder of this paper is structured as follows: Section II contains a more detailed discussion of the related work. In Section III, we present our genetic algorithm for the construction of Bayesian networks. Section IV describes our idea for reducing attribute disclosure risks via customized Bayesian networks. Our experiments can be found in Section V, and Section VI contains conclusions and discusses future work.

II. Related Work

The process of data synthetization generally consists of two steps: (1) Learn the properties of the original dataset, i.e. the distribution of its attributes and the correlations between them, and store them in a model. (2) Use the model to generate synthetic samples, e.g. by drawing random values from the learned distributions. The first step is commonly referred to as 'data description', while the second step is referred to as 'data generation'.

In this paper, we are particularly interested in using *Bayesian networks* [14] for describing correlations between attributes of the original dataset. A Bayesian network on a dataset D is a directed, acyclic graph that compactly describes the high dimensional probability distributions on the input data. The attributes are represented as nodes in the graph, and a conditional dependence between any two attributes is represented as an edge between the respective nodes. The construction of a Bayesian network that best describes the dependencies on the input data is a difficult task and has been shown to be NP-complete [15]. In [8], a greedy algorithm (*GreedyBayes*) has been presented that is also used in the DataSynthesizer [5]. While the returned networks may be used to generate synthetic data with high utility, the procedure is still too slow for datasets with higher dimensions. As our first contribution, we solve this task efficiently using a *genetic algorithm* [16]. In general, genetic algorithms solve an optimization problem by representing the search space by a number of individuals (the "population") that represent possible solutions. Each individual is represented by so-called "chromosomes" that reflect its properties. The best individuals (with regard to some fitness function) are selected as parents for the next generation, which is obtained by applying certain changes (the "crossovers" and "mutations") to the chromosomes. It has been shown that genetic algorithms can be used to construct Bayesian networks (e.g., [17], [18]). Section III contains a complete description of our own algorithm.

The second contribution of this paper is related to decreasing disclosure risks on synthetic data; as discussed above, we focus on attribute disclosure risks. The concept of Correct Attribution Probability (CAP) has been introduced in [19] and elaborated on in [9] and [10], where it has been generalized to GCAP. For assessing attribute disclosure risk, GCAP assumes that the attacker knows the values of a set of attributes (the 'quasi-identifiers', forming a specific 'key') of an individual in the original dataset, and wants to learn the respective value of some sensitive attribute. Such a scenario has also been studied in [20] (although, based on another definition, the authors speak of identity instead of attribute disclosure). Consider a dataset consisting of micro-data with n records representing individuals and a number of attributes. For $j \in \{1, \dots, n\}$, let $K_{O,j}$ be the vector representing the values of the key attributes of the j-th record in the original dataset O, and let $T_{O,j}$ be the corresponding value of the sensitive attribute. Similarly, we define $K_{S,j}$ and $T_{S,j}$ for the synthetic dataset S. The basic idea is that the attacker is assumed to search for all records in S that match the key attribute values known by them. We call this subset of data points the *equivalence class* of $K_{O,j}$ in S. Inside this class, they then calculate the distribution of the occurring values of the sensitive attribute. $\mathrm{GCAP}_{S,j}$ then corresponds to the proportion of the actual sensitive value $T_{O,j}$ in this equivalence class. In this sense, $\mathrm{GCAP}_{S,j}$ measures the risk of disclosure of this information about the individual represented by the j-th record in the original data. The difference between GCAP and the original notion CAP is the handling of cases where the vector $K_{O,j}$ does not occur in the synthetic dataset. While the original notion is undefined, GCAP then considers vectors $K_{S,j}$ that match at least some of the values of the known key.

For a distance metric[3] Δ, the GCAP score for record j in the original dataset is the empirical probability of its sensitive value given its key attribute values, that is

$$\mathrm{GCAP}_{S,j} := \frac{\sum_{i=1}^{n} [T_{S,i} = T_{O,j} \wedge \Delta(K_{S,i}, K_{O,j}) = \rho]}{\sum_{i=1}^{n} [\Delta(K_{S,i}, K_{O,j}) = \rho]} \quad (1)$$

where $\rho := \min \{r \mid \exists i \in \{1, \dots, n\} : \Delta(K_{S,i}, K_{O,j}) = r\}$ and $[\cdot]$ is the Iverson bracket. The purpose of ρ is to check for records in the synthetic dataset that match at least some values of $K_{O,j}$ in cases where no complete match exists.[4]

A central idea of [10] is to consider the situation of the attacker as classification problem. Given the synthetic dataset and some background knowledge in form of values of key attributes of an individual in the original data, obtain a prediction for said individual's sensitive attribute. Using majority voting, the GCAP approach may be turned into a classifier that is closely related to the fixed-radius nearest neighbor algorithm, and the average score of $\mathrm{GCAP}_{S,j}$ over all individuals j yields its accuracy. In our experiments in Section V, we will use GCAP together with a number of other classification algorithms to evaluate the attribute disclosure risk on the considered synthetic datasets.

[3]For categorical data, we can use the Hamming distance.

[4]The original notion CAP is equal to GCAP with fixed $\rho = 0$.

Finally, it is worth noting that the attacker model of GCAP is closely related to that of traditional privacy concepts such as k-anonymity and l-diversity (see [1], [21]). A dataset has the k-anonymity property if, for every combination of quasi-identifiers occurring in the data, the corresponding equivalence class consists of at least k elements. A dataset has the l-diversity property if, in every equivalence class, the sensitive variable takes on at least l distinct values. The techniques related to these notions try to reduce an attacker's abilities to deduce the value of a sensitive attribute on the original dataset. In Section IV, we will present an approach to do the same on a synthetic dataset.

III. A GENETIC ALGORITHM FOR BAYESIAN NETWORKS

In the original implementation of PrivBayes [8] and subsequently also the DataSynthesizer [5], a greedy algorithm is employed to learn a Bayesian network on the input dataset. First, a random attribute f_1 is chosen as root of the network. The next attribute f_2 is chosen based on the goal of maximizing the mutual information between f_1 and f_2. We then keep adding attributes to the network in this manner, always considering all possible combinations of already added attributes as parents in order to maximize mutual information. The preset parameter k defines the maximum number of parents for each node. Thus, the number of possible parent combinations for attribute f_i we have to consider is given by the binomial coefficient $\binom{i-1}{k}$. Summing over all iterations i and possible choices for f_i in each of them, the total number of considered combinations may be bounded by $d\binom{d+1}{k+1}$, where d is the number of attributes in the dataset. In addition, the number of samples influences the computational cost for computing mutual information between child nodes and the joint distribution of its parents. For increasing k or large dimensions of the dataset, the computational cost thus quickly becomes substantial. In Section II, we discussed the difficulty of constructing high quality Bayesian networks on datasets with a large number of attributes. If the input dataset has hundreds to thousands of attributes, it may make more sense to use other synthesizers with different approaches, e.g. the *Synthetic Data Vault* [6]. On the other hand, the DataSynthesizer with its Bayesian network approach has performed almost always better in evaluations on datasets with a small to moderate number of attributes ([11], [12]). To improve the efficiency of the data description step on high-dimensional datasets, we consider an alternative to the greedy approach in the form of a genetic algorithm.

The individuals of our population each correspond to a possible Bayesian network. We will measure the utility of these networks by focusing on *pairwise* mutual information, and select the best members of each generation for the application of crossovers and mutations. In previous publications on using genetic algorithms for the construction of Bayesian networks (e.g., [18]), it has been demonstrated that the performance can be improved by representing each individual of the population by two chromosomes, the "ordering" and the "connectivity" chromosome. The ordering chromosome encodes the order by which the attributes of the dataset are added to the Bayesian network. The connectivity chromosome encodes the parent nodes for each of these attributes. In [18], the connectivity chromosome is a binary-coded upper triangular matrix. One of the main problems is that only those nodes can be selected as parents of a certain attribute that appear prior to said attribute in the ordering chromosome. Therefore, after applying mutations and crossovers to the ordering chromosome, the authors also changed the connectivity chromosome accordingly to make sure that the individual still represents a valid Bayesian network structure.

While we will use the same dual chromosome approach as discussed above, the main difference between our algorithm and the method in [18] is that our connectivity chromosome is not a binary-encoded matrix, but a list of sets of the selected parents for each attribute. In our experiments in Section V, we will see that this straight-forward approach is well-suited for the synthetization of the considered datasets. Compared to the performance of the greedy algorithm, it constitutes a substantial improvement in efficiency.

Again, let k be the maximum number of parents for each node in the network. The genome of each individual consists of the following two chromosomes:

- The ordering chromosome $\mathcal{O}$: A permutation of the attributes $f_1, \ldots, f_d$ of the input dataset
- The connectivity chromosome $\mathcal{C}$: A list of d sets, where the i-th set corresponds to the attribute f_i. Each set contains exactly k attributes and represents the possible parents of f_i in the network.

We point out that, in contrast to [18], the connectivity chromosome and the ordering chromosome are notationally independent. The i-th set in the connectivity chromosome always refers to f_i, no matter the place of f_i in the ordering chromosome. In addition, our algorithm has four parameters:

- The population size P: The number of individuals in each generation, set to 200 by default
- The selection pressure S: The number of individuals with the highest fitness that are selected for creating the next generation, set to 10 by default
- The mutation rate r: A number in $[0, 1]$, set to $1/d$ by default
- The iterations e: The number of overall generations

We then run the procedure **Initialization** to initialize the first generation of P individuals.

The next step is to evaluate the individuals and rank them based on the suitability of the corresponding Bayesian network. We determine the fitness of each individual by computing the total sum of all pairwise mutual information scores[5] of all nodes and their valid parents in the network[6]. The most efficient way to determine these sums during the procedure is to precompute the pairwise mutual information

[5]We used sklearn's function "mutual_info_score" (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mutual_info_score.html)

[6]In the conversion of the chromosomes to a Bayesian network, we have to make sure that we obtain a valid network structure. If an attribute in C_j does not precede f_j in $\mathcal{O}$, we simply ignore it and do not add it as parent of f_j.

Initialization

```
1: for individual $i$ from 1 to $P$ do
2:     Generate a random permutation $\mathcal{O}$ of $f_1, \ldots, f_d$
3:     Let $\mathcal{O}$ be the ordering chromosome of individual $i$
4:     for $j = 1, \ldots, d$ do
5:         Let $F$ be the set of attributes that appear prior to $f_j$ in $\mathcal{O}$
6:         If $|F| \geq k$, let $C_j$ be a set of $k$ distinct random elements of $F$
7:         If $|F| < k$, let $C_j$ contain all elements of $F$ and $k - |F|$ distinct random attributes not in $F$
8:     Let $\mathcal{C} = [C_1, \ldots, C_d]$ be the connectivity chromosome of individual $i$
9:     Save the individual $i = (\mathcal{O}, \mathcal{C})$
```

scores for all possible pairs of attributes and store them.[7] Subsequently, we find the S individuals with the highest fitness and apply the following mutation and crossover operations to them. We first consider the operation for the crossover between two distinct individuals $i_1 = (\mathcal{O}_1, \mathcal{C}_1)$ and $i_2 = (\mathcal{O}_2, \mathcal{C}_2)$.

Crossover(i_1, i_2)

```
1: Generate a random integer $j$ in $[0, d]$
2: Return $i = (\mathcal{O}_1, \mathcal{C})$, where $\mathcal{C}$ contains the first $j$ sets of $\mathcal{C}_1$ and the last $d - j$ sets of $\mathcal{C}_2$
```

In contrast to [18], our crossover operation mostly concerns the connectivity chromosome and not the ordering chromosomes $\mathcal{O}_1$ and $\mathcal{O}_2$. We found it sufficient to reach a suitable ordering of the attributes by standard mutations. Hence, our first mutation is a flip in the permutation of the ordering chromosome $\mathcal{O}$ of an individual $i = (\mathcal{O}, \mathcal{C})$.

Order Flip(i)

```
1: for $f$ in $\mathcal{O}$ do
2:     Generate a random number $x$ in $[0, 1]$
3:     if $x < r$ then
4:         Generate a random integer $j$ in $[1, d]$
5:         Flip $f$ with $f_j$ in $\mathcal{O}$
6: Return $i$ with updated ordering chromosome $\mathcal{O}$
```

The second mutation is applied to the elements of the sets in the connectivity chromosome $\mathcal{C} = [C_1, \ldots, C_d]$ of an individual $i = (\mathcal{O}, \mathcal{C})$. It will be the last operation applied after **Crossover** and **Order Flip** in the construction of elements of the following generation. Besides the standard random swaps of elements in the list of parents of each node (see Steps 7 to 12 in **Swap**), we will also repair possible invalid network structures that may have resulted from changes to the ordering chromosome. In the Steps 2 to 6, we swap elements in the sets of parents that do not precede the corresponding attribute in $\mathcal{O}$ against random elements that do.

[7]The precomputation of these scores is the only part where our technique interacts with the input dataset.

Swap(i)

```
1: for $j = 1, \ldots, d$ do
2:     Let $G$ be the set of attributes in $C_j$ that appear later than $f_j$ in $\mathcal{O}$
3:     for $g$ in $G$ do
4:         Let $F$ be the set of attributes that appear prior to $f_j$ in $\mathcal{O}$ *and* are not already in $C_j$
5:         if $F \neq \emptyset$ then
6:             Swap $g$ with a random element in $F$
7:     for $c$ in $C_j$ do
8:         Generate a random number $x$ in $[0, 1]$
9:         if $x < r$ then
10:            Let $F$ be the set of attributes that appear prior to $f_j$ in $\mathcal{O}$ *and* are not already in $C_j$
11:            if $F \neq \emptyset$ then
12:                Swap $c$ with a random element in $F$
13: Return $i$ with updated connectivity chromosome $\mathcal{C}$
```

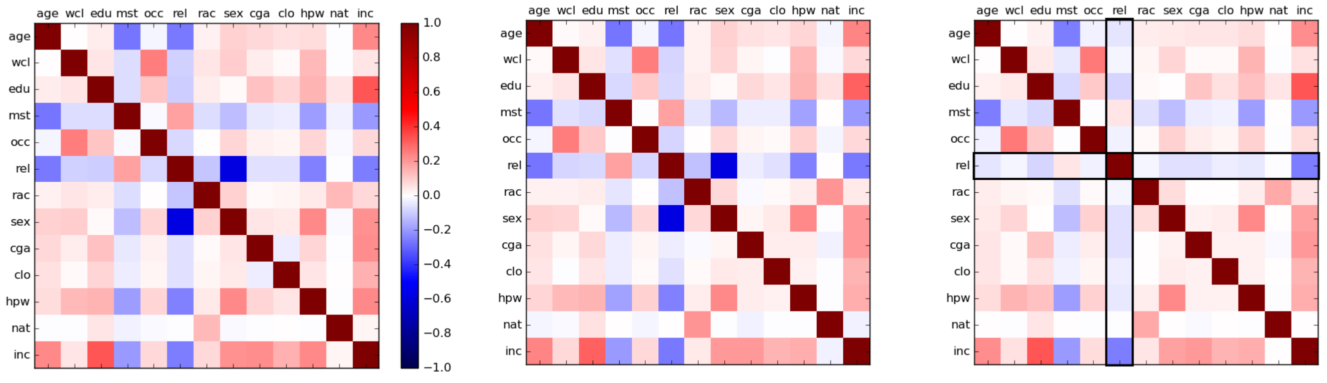
In order to create the next generations $\mathcal{G}$ of P individuals, we perform the procedure **Next Generation**, where we start by adding the S individuals with the highest fitness from the previous generation to $\mathcal{G}$ without altering them (often called "elitism"). After completion of the procedure, we again evaluate and rank the individuals in $\mathcal{G}$ to find the S fittest ones for obtaining the next generation. We repeat this process e times. In the last generation, we choose the best network (with the highest fitness) as the output. The results of comparing this algorithm to the greedy approach are presented in Section V-A.

Next Generation

```
1: Let $\mathcal{E}$ be the set of the $S$ fittest individuals from the previous generation, and let $\mathcal{G} = \mathcal{E}$.
2: while $|\mathcal{G}| < P$ do
3:     Choose a random individual $i$ in $\mathcal{E}$
4:     Generate a random number $x$ in $[0, 1]$
5:     if $x < r$ then
6:         Choose a random individual $i'$ in $\mathcal{E}$
7:         $i \leftarrow$ Crossover$(i, i')$
8:     $i \leftarrow$ Order Flip$(i)$
9:     $i \leftarrow$ Swap$(i)$
10:    Add $i$ to $\mathcal{G}$
```

IV. Custom Bayesian networks for specific tasks

While the main purpose of the method presented in Section III is to improve the efficiency of synthetization based on Bayesian networks, our second contribution is an adaptation to further reduce the privacy risk on synthetic data. Data synthesizers are often intended to generate datasets for general purposes, and do not assume a particular order or quality of attributes. For this reason, they deal with them in an arbitrary or random order. As an example, the original algorithm of PrivBayes, as implemented in the DataSynthesizer, builds the



(a) Original data (b) DS with genetic algorithm (c) Custom DS with genetic algorithm

Fig. 1: Heatmaps showing the Pearson correlation coefficients on Adult Census. In Figure 1c, the sensitive attribute is 'relationship' (rel) and the target attribute is 'income' (inc). The correlations between 'relationship' and other attributes besides 'income' are considerably decreased.

model by taking a random attribute as root of a Bayesian network and adding parents in a greedy manner to maximize mutual information. In contrast to the concept of k-anonymity and the identification of quasi-identifiers, data synthesizers usually do not require any information about the attributes, their privacy implications and their relation to each other.

Imagine a table with a column consisting of the target attribute of a particular machine learning task, and various other columns for the predictors (i.e., input variables to the model), including potentially sensitive data. If we know which attributes are more sensitive before employing data synthetization, we may modify the procedure to optimize the generation of synthetic data. The goal is to provide a method for keeping the utility of the data for the training of classifiers as high as possible, while simultaneously reducing the disclosure risk (e.g. GCAP, cf. Section II) of particularly sensitive attributes among the predictors. The proposed approach consists of two steps. The first step is to divide the attributes into three subsets: the nonsensitive predictors, a sensitive predictor and the target attribute. In the second step, we use the synthesizer to generate synthetic records respecting certain restrictions for the groups of attributes. The sensitive attribute should be sampled based on its own distribution and the conditional probabilities to the target attribute. In the group of nonsensitive predictors (to which the quasi-identifiers belong), we may take all correlations into account *except* those to the sensitive attribute. Finally, the target attribute may be sampled without any restrictions. By doing so, we pursue two goals. First, we want to detach the sensitive attribute from the nonsensitive attributes by generating it without the corresponding correlations. Second, we want to keep the dependency between the sensitive and the target attribute such that the relation between them can still be learned and analyzed. The key for achieving these goals is to find the right structure of the Bayesian network used for the data generation step. We start by adding the target attribute as root to the network, and adding the sensitive attribute as the second node. As a result, the target attribute will be the only parent node of the sensitive attribute. Subsequently, we add all other attributes (i.e., nonsensitive

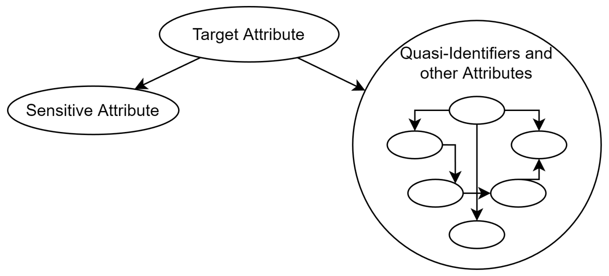


Fig. 2: Proposed custom Bayesian network structure

predictors) in the usual manner to the network, with one important restriction: the sensitive attribute is not allowed as parent for any of these other attributes. We thereby simulate a conditional independence between the sensitive and other predictors. The general structure of the resulting network is displayed in Figure 2.

We may easily modify the genetic algorithm introduced in Section III to achieve this behavior[8]. We just fix the target attribute as the first and the sensitive attribute as the second component in the ordering chromosome $\mathcal{O}$. In addition, we add a restriction that the sensitive attribute is not a permissible parent to any of the nonsensitive attributes, similar to the restriction we already have for attributes that appear later in the ordering chromosome. Besides that, we run the genetic algorithm as explained. The resulting network then looks like discussed above. In order to get a better understanding of the consequences, let us take a look at Figure 1, which shows pairwise correlation coefficients for the attributes from the Adult Census dataset (one of the datasets used in our experiments, see Table I). Figure 1b shows the correlation coefficients of the synthetic dataset without any custom modifications. One observes that the heatmap looks very similar to Figure 1a, which shows the correlations on the original data. We then generated synthetic data based on a custom Bayesian network with the target attribute 'income' and the sensitive attribute

[8]Similar modifications may be applied to the original (greedy) implementation of the DataSynthesizer.

‘relationship’. The resulting heatmap is shown in Figure 1c. It again looks similar to the original, but all correlations between ‘relationship’ and the nonsensitive predictors are considerably decreased. Simultaneously, the negative correlation to ‘income’ is preserved, as we desired.

In our experiment in Section V-B, we evaluate both the resulting decrease in privacy risks, as well as the influence on the utility of the data for machine learning.

V. Evaluation

Section V-A presents the results on comparing the runtime complexity between the genetic and the greedy algorithm for constructing Bayesian networks, which was introduced in Section III. We also consider the utility of the networks resulting from both techniques, measured by the effectiveness of machine learning classifiers. Section V-B evaluates the custom Bayesian networks described in Section IV.

TABLE I: Dataset Characteristics

Dataset	# Features	# Instances	# Classes
Contraceptive Method Choice[9]	9	1,437	3
Adult[10] (Census Income[11])	15	48,842	2
Credit Card Fraud[12]	31	284,807	2

We conduct our experiments on three publicly available datasets. To study and compare the scalability of the approaches for constructing Bayesian networks in Section V-A, we selected datasets of different sizes and dimensions. In addition, we will use the classification tasks on the three datasets for our utility assessments in both Sections V-A and V-B. We apply five machine learning algorithms, namely Naïve Bayes, Support Vector Machines, k-Nearest Neighbors, Random Forests and Logistic Regression. All classifiers are implemented in the Python sklearn package[13], and we used standard parameters. Our goal was not to reach maximum performance for the considered tasks, but to achieve comparability between models trained on original data and models trained on synthetic data.

A. Genetic and Greedy Algorithm for Bayesian Networks

Let us now consider the performance of the two approaches for constructing Bayesian networks, and how they affect the utility of the resulting synthetic dataset for machine learning. For each dataset in Table I and fixed choices of the Bayesian network degree $k \in \{1, 2, 3, 4\}$ (maximum number of parents of each node), we performed the following procedure:

1) We deleted particular columns (as part of standard feature cleaning), and generated a random split of the dataset into training (80%) and test data (20%). For the Adult Census dataset, we used the pre-defined split.
2) We applied the original implementation (greedy algorithm) and our adaptation (genetic algorithm[14]) of the DataSynthesizer[15] to the original training dataset to generate synthetic training data.
3) On all training datasets and the test dataset, we applied label encoding and, if necessary, one-hot encoding. We then used sklearn’s StandardScaler for feature scaling.
4) We fitted the machine learning models to the training datasets and predicted the results on the test dataset.

We first compare the runtimes[16] of generating the Bayesian networks in Step 2. For each dataset, we show a table with the runtime (in seconds) of both the data description and the data generation step. In addition, the results are visualized in a bar chart, where the light-colored part of the bars represents the data description and the dark-colored part represents the data generation step.

Let us start by considering the Contraceptive Method Choice dataset (Table II). Despite the fact that the dimensions of this dataset are comparably small, we already observe a substantially improved efficiency of the genetic algorithm compared to the greedy algorithm. Interestingly, the application of the technique in Section III does not only improve the runtime of the data description step (where the Bayesian network is created), but also of the data generation step of the synthesizer. These differences are even more pronounced for the Adult Census dataset (Table III). The structure of the Bayesian network seems to have a profound influence on the runtime of data generation.[17] In general, however, the complexity of this part of the procedure grows exponentially with regards to the values of k. For $k = 4$, the data generation step is much more expensive than the data description step for both methods. The Credit Card Fraud dataset (Table IV) with its 31 attributes and almost 300,000 samples is the largest of the considered datasets and allows us to evaluate the scalability of both approaches to larger instances. On this dataset, we may thus observe the largest differences between the methods. Due to time issues in the computation of the case $k = 4$, we have only considered the values $k = 1, 2, 3$. Considering the case $k = 3$, the genetic algorithm constructs the Bayesian network in less than 10 minutes, while the greedy algorithm takes almost 12 hours. Still, the data generation step takes almost 1.5 hours for both methods.

To summarize, the genetic algorithm appears to perform well for increasing data dimensions. In particular, it shows only a linear increase in the runtime complexity of the data

[9]https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

[10]https://archive.ics.uci.edu/ml/datasets/Adult

[11]https://archive.ics.uci.edu/ml/datasets/census+income

[12]https://www.kaggle.com/mlg-ulb/creditcardfraud

[13]https://scikit-learn.org/stable

[14]We applied it with the standard choices of the parameters P, S and r. On the Credit Card Fraud dataset, we used $e = 2000$ to make sure that the solution converges. On the other two datasets, we used $e = 400$.

[15]The DS allows to set a differential privacy parameter ε. As mentioned in the introduction, we ran our experiments without differential privacy.

[16]All computations have been conducted on a standard laptop with AMD Ryzen 7 5800H processor (8-core 3.2 GHz) and 16GB RAM.

[17]The algorithm samples from the conditional distributions by looping through all combinations of unique values of the parent attributes for each node. On categorical data, the genetic algorithm tends to include parents with a lower number of unique values than the greedy approach, leading to faster completion of the loop. On the predominantly continuous attributes of the Credit Card Fraud, there appears to be no significant difference (see Table IV).

TABLE II: Contraceptive Method Choice. Runtime (in seconds) shown as: Runtime Data Description + Runtime Data Generation

	k=1	k=2	k=3	k=4
DS	31.92 + 0.06	33.94 + 0.62	35.41 + 3.29	39.03 + 15.18
DS GA	5.62 + 0.06	7.81 + 0.38	9.51 + 1.73	12.78 + 6.13

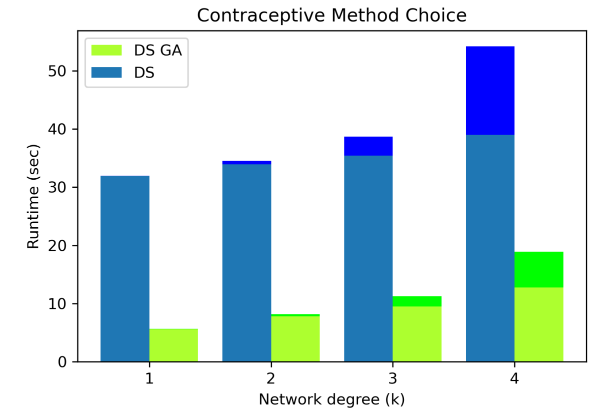
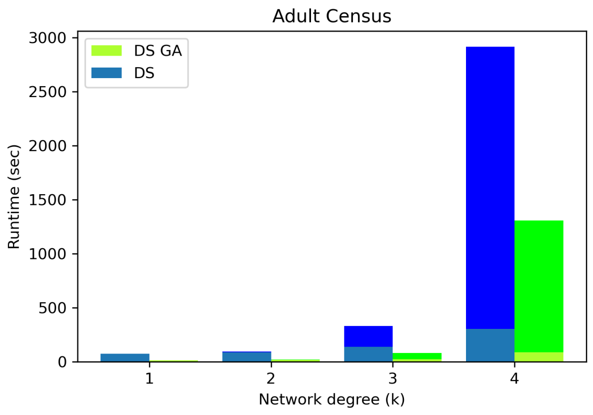


TABLE III: Adult Census. Runtime

	k=1	k=2	k=3	k=4
DS	73.62 + 0.62	89.21 + 7.01	139.57 + 190.46	303.90 + 2610.69
DS GA	14.17 + 0.63	17.01 + 4.32	23.12 + 58.69	86.36 + 1219.53



description step for increasing values of k, whereas the experiment on Credit Card Fraud demonstrated that the runtime of the greedy algorithm is quite sensitive to this parameter.

While improving the runtime complexity was our main goal, it is also crucial to test if the resulting Bayesian networks are of satisfying quality. We thus evaluated utility scores of the classification machine learning tasks on the considered datasets. On the Contraceptive Method Choice dataset, we ran the Steps 1-4 ten times with different random splits to compare the performance of the machine learning classifiers. On the Adult Census dataset, we used the pre-defined split. On the Credit Card Fraud dataset, we considered one random split. As we use four different values of k for each of the two methods for generating synthetic data, and we also evaluate the original data as baseline, we learn our models on a total of nine different training sets (seven on the Credit Card Fraud, since we skipped $k = 4$).

TABLE IV: Credit Card Fraud. Runtime

	k=1	k=2	k=3
DS	3335.86 + 16.54	8296.80 + 170.55	42976.82 + 4463.27
DS GA	456.21 + 16.67	500.10 + 167.37	577.05 + 4423.71

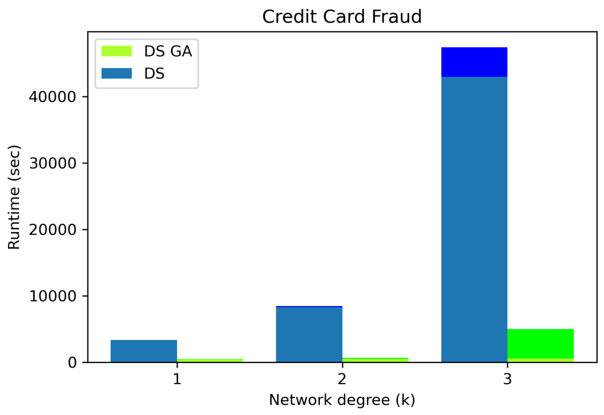
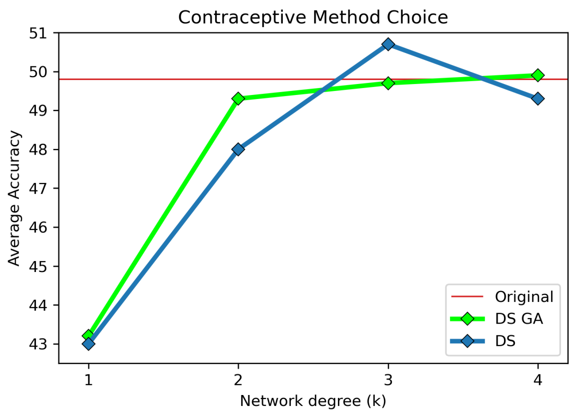


TABLE V: Contraceptive Method Choice. ML Utility shown as: Mean Accuracy Scores ± Standard Deviation (Ten random splits)

	NB	SV	KN	RF	LR	Avg
Original	44.9±2.7	54.6±2.4	48.2±2.8	50.6±2.6	50.7±1.7	49.8
1: DS	45.9±3.2	46.3±4.5	39.8±2.8	40.8±2.8	42.2±1.4	43.0
1: DS GA	42.7±3.4	46.5±2.8	41.8±3.0	43.3±2.7	41.8±4.9	43.2
2: DS	48.5±2.7	51.7±3.6	42.7±1.7	47.5±2.4	49.6±3.4	48.0
2: DS GA	47.5±2.6	53.9±2.7	45.6±2.4	49.0±2.9	50.3±2.5	49.3
3: DS	47.2±2.0	54.2±2.3	47.7±2.5	51.8±3.5	52.5±2.7	50.7
3: DS GA	47.1±2.5	54.0±3.3	46.3±3.2	49.3±1.3	51.8±1.6	49.7
4: DS	45.5±1.9	53.8±2.2	45.6±2.8	50.4±2.5	51.4±1.8	49.3
4: DS GA	45.6±3.0	55.1±2.6	46.6±1.7	51.0±1.5	51.2±2.2	49.9

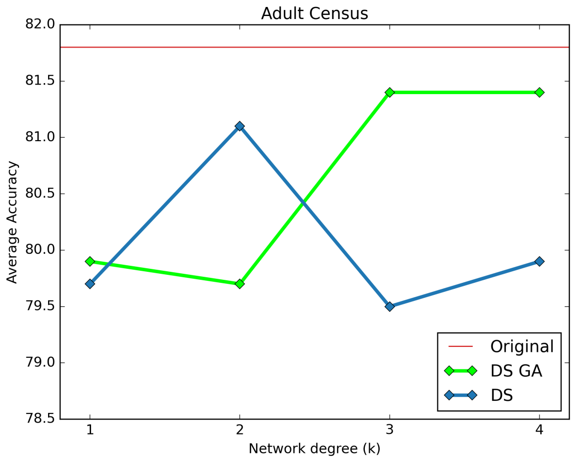


Let us start with the Contraceptive Method Choice dataset shown in Table V. We report the mean accuracy scores (in percent) of the five classification algorithms for predicting the correct value of the target attribute (Contraception method: No-use, Long-term, Short-term) over the ten random splits of the dataset together with the standard deviation. In addition, we report the average of the mean accuracy scores in the last column of the table. This value is also visualized in a plot below the table. The original implementation of the

DataSynthesizer shows a high performance for $k = 3$ and even exceeds the scores on the original dataset by a substantial margin. However, our adaptation achieves higher average scores for $k = 1, 2, 4$. For $k = 2, 3, 4$, the DataSynthesizer applied with a genetic algorithm shows an average accuracy score that is very close to the one obtained on the original dataset. In any case, we may conclude that the application of the genetic algorithm is very much suitable for this particular task.

TABLE VI: Adult Census. ML Utility (Public split)

	NB	SV	KN	RF	LR	Avg
Original	79.7	83.2	81.5	82.2	82.3	81.8
1: DS	80.2	80.8	77.9	79.0	80.5	79.7
1: DS GA	80.2	80.8	78.6	79.3	80.7	79.9
2: DS	80.8	82.4	80.3	80.2	81.8	81.1
2: DS GA	80.1	80.9	77.4	79.5	80.6	79.7
3: DS	80.4	81.0	77.9	78.0	80.1	79.5
3: DS GA	80.2	83.1	80.5	81.0	82.0	81.4
4: DS	80.0	81.3	78.1	79.8	80.4	79.9
4: DS GA	79.8	83.2	81.3	81.1	81.8	81.4



On the Adult Census dataset (Table VI), the task is to predict whether the yearly income of a person is below or above $\$50k$. We can see that the genetic algorithm again achieves a higher average accuracy score on three of four values of k. In general, both synthesizers are not as close to the original performance as they have been on the Contraceptive Method Choice dataset. However, our adaptation comes quite close for $k = 3$ and $k = 4$. Compared to the greedy algorithm, we again conclude that the performance is more than satisfactory.

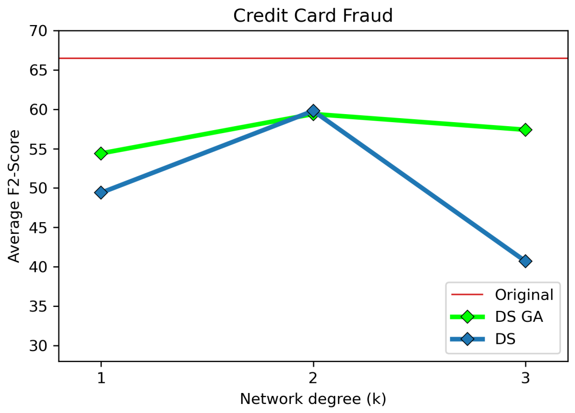
Finally, let us consider the Credit Card Fraud dataset (Table VII). The task is to detect frauds, and the target attribute is highly imbalanced. We report the results as confusion matrices of the form

$$\begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix},$$

where TP means true positive, FP means false positive, TN means true negative and FN means false negative. **Precision** is defined as the ratio of true positives (correctly identified

TABLE VII: Credit Card Fraud. ML Utility via Confusion Matrices (One random split)

	NB		SV		KN		RF		LR		Avg F2
Original	55626	1224	56840	10	56845	5	56848	2	56836	14	66.5
	26	86	33	79	25	87	32	80	26	86	
1: DS	55218	1632	56846	4	56846	4	56828	22	56783	67	49.4
	16	96	58	54	55	57	64	48	29	83	
1: DS GA	55132	1718	56846	4	56845	5	56830	20	56784	66	54.4
	17	95	51	61	53	59	43	69	29	83	
2: DS	55298	1552	56841	9	56842	8	56842	8	56803	47	59.8
	20	92	42	70	39	73	37	75	30	82	
2: DS GA	55368	1482	56844	6	56842	8	56836	14	56770	80	59.4
	24	88	50	62	34	78	33	79	27	85	
3: DS	55205	1645	56850	0	56837	13	56833	17	56774	76	40.7
	23	89	112	0	43	69	58	54	32	80	
3: DS GA	55376	1474	56846	4	56842	8	56838	12	56821	29	57.4
	25	87	48	64	43	69	42	70	32	80	



frauds) to all data points predicted as frauds, i.e. how many of the cases the model has *predicted* to be frauds are actually such. **Recall** indicates how many of the frauds have been identified by the model, given as the ratio of the true positives to all cases that have been *labeled* as fraud. Each of these measures alone is not representative, as it is rather easy to optimize one of them, but difficult to have both take high values at the same time. As summarizing measure, the last column of the tables thus presents the average **F2 score** (in percent), a weighted harmonic mean of precision and recall that weighs recall higher than precision. It appeared suitable in our application, where it is likely more important to identify most of the frauds, and a certain amount of false positives can be tolerated. While dealing with imbalanced datasets is usually more difficult for data synthesizers (due to the focus on global, not local properties), the results are again not too far from those algorithms trained on original data. Considering the average F2 scores, the genetic algorithm improves upon the greedy algorithm for $k = 1$ and $k = 3$, while the greedy algorithm shows a slightly better performance on $k = 2$.

Overall, our adaptation using the genetic algorithm generates synthetic data with high utility for machine learning on the considered datasets. The performance for higher values of k seems to be more stable than on the greedy algorithm, which has shown a rather large drop of its scores for either $k = 3$ or $k = 4$ on each of the three datasets. Together with the reduction of the runtime complexity, in particular for higher k

and for larger datasets, we conclude that the generation of the Bayesian network with a genetic instead of a greedy algorithm may be preferable in many practical scenarios.

B. Disclosure Risks and Utility on Custom Bayesian Networks

Our evaluation of the approach discussed in Section IV considers two aspects. The first is to verify if the attribute disclosure risk is indeed reduced by decreasing particular pairwise correlations. For this, we will conduct similar experiments as those in [10]. The second is to repeat the utility experiments conducted in Section V-A on the custom Bayesian networks to evaluate possible performance losses.

Let us first take a look at the experiments on the attribute disclosure risks. In Section II, we have discussed the GCAP measure and the attacker's classification problem. We define an attack scenario as a set of quasi-identifiers QI together with a sensitive attribute. Our assumption is that an attacker knows at least some of their victim's values of the QI (the 'key'), and tries to find the value of the sensitive attribute in the original dataset by studying the synthetic dataset. On the Contraceptive Method Choice and the Adult Census dataset, we analyze one scenario each.

TABLE VIII: Disclosure Risk Scenarios

	Quasi Identifiers	Sensitive Attr.
1) Contraceptive Method	age, education, number of children, religion, now working?	education of husband
2) Adult Census	age, workclass, occupation, race, sex	relationship

The idea behind the scenario on the Contraceptive Method dataset is to investigate the possibility of gaining information about the husbands' education (1=low, 2, 3, 4=high) based solely on knowledge about the wives[18]. In the scenario for Adult Census, we assume that the adversary wants to learn the relationship status (Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried) of a person.

Let O be the original table (of either dataset). For both Scenarios 1) and 2) and a certain key length ℓ, we performed the following procedure.

1) Generate two synthesized versions of O of equal length:
   - DataSynthesizer[19] with ordinary Bayesian network
   - DataSynthesizer with custom Bayesian network
     sensitive attribute: as in the scenario
     target attribute: 'method' for 1), 'income' for 2) (as in the utility evaluation in Section V-A)
2) Compute all ℓ-element subsets of the quasi identifiers QI of the respective scenario. Each subset corresponds to an attribute key used in the following step.
3) For each synthetic dataset S, for each attribute key K and the sensitive attribute T of the scenario:
   - Train GCAP and other ML models on $O\big|_{K,T}$ and $S\big|_{K,T}$, the datasets that results from omitting all attributes but T and those in the key K.
   - Compute a prediction for the sensitive attribute of all original records (meaning that $O\big|_{K,T}$ is the dataset on which all models are "tested").
   - Compute the overall accuracy score on $O\big|_{K,T}$.

[18]This scenario has also been studied in the original publication [10].

[19]We used the genetic algorithm (Section III) with standard choices for P, S, r and $e = 400$. Also, we set $k = 4$ as degree for the network.

TABLE IX: Contraceptive Method Choice. Attribute Disclosure Risk for Scenario 1 (Keylength 4)

	GCAP	NB	SV	KN	RF	LR	Avg
Original	77.8±7.0	64.2±1.7	65.0±1.5	69.4±3.5	77.0±6.6	64.4±1.6	69.6
DS	65.6±4.0	63.9±1.6	64.0±1.4	64.2±4.7	65.7±4.2	64.4±1.5	64.6
DSc	55.0±3.4	61.2±0.2	61.0±0.0	54.4±1.7	54.9±3.2	61.0±0.0	57.9

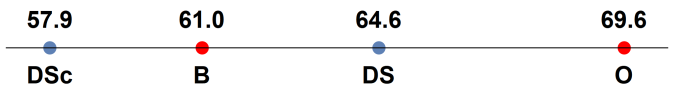
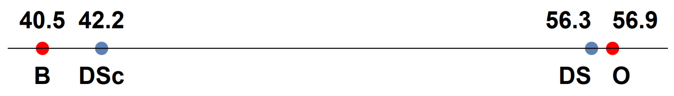


TABLE X: Adult Census. Attribute Disclosure Risk for Scenario 2 (Keylength 5)

	GCAP	NB	SV	KN	RF	LR	Avg
Original	60.1	45.4	59.8	56.8	60.1	59.3	56.9
DS	59.7	45.4	59.5	54.4	59.7	59.3	56.3
DSc	42.6	43.5	41.3	42.2	42.5	41.3	42.2



For Scenario 1) (Table IX), we consider a set of five QIs and assume that the attacker knows at least four of them ($\ell = 4$), as it was also considered in [10]. For Scenario 2) (Table X), we have $|QI| = 5$ and assume that an attacker knows all of them ($\ell = 5$). We applied the same classification algorithms that we have already used in Section V-A. In addition, we computed the GCAP score (eq. (1)). In each cell of Table IX, we display the mean accuracy scores over all assumed attribute keys, i.e., the five 4-element subsets of QI. The cells of Table X just show the results of the complete attribute key of length 5. The last column of the tables present the average score of the mean accuracies in the respective row[20]. These scores are also visualized in the figures below the tables. Here, O denotes the score of the original dataset, DS the score of the DataSynthesizer with ordinary Bayesian network and DSc the score of our customization. In addition, we have added the score B to the figure, which denotes the prediction baseline for the sensitive attribute that is obtained by just applying a dummy ("zero-rule") classifier that always predicts the most common value. For the attribute 'education of husband' in the first scenario, this value is 61.0%. For 'relationship' in the second scenario, it is 40.5%.

On the original dataset, an attacker would be able to use their knowledge of the quasi-identifying values in both scenarios to make accurate predictions for the sensitive attribute.

[20]It is also interesting to consider the maximum accuracy scores as some sort of upper bound for the disclosure risk. However, an attacker is unlikely to know which classifier works best on the dataset they are analyzing, so we are reporting the average performance of several distinct algorithms.

TABLE XI: Contraceptive Method Choice. ML Utility on customized synthetic data

	NB	SV	KN	RF	LR	Avg
DS	45.6±3.0	55.1±2.6	46.6±1.7	51.0±1.5	51.2±2.2	49.9
DSc	46.0±2.7	54.0±1.9	47.2±2.0	51.0±3.0	50.8±2.2	49.8

TABLE XII: Adult Census. ML Utility on customized synthetic data

	NB	SV	KN	RF	LR	Avg
DS	79.8	83.2	81.3	81.1	81.8	81.4
DSc	79.7	83.1	80.4	80.2	80.8	80.8

The GCAP score and the performance of Random Forest are particularly high. If the attacker has only access to the synthetic dataset DS, their abilities to predict the correct value are already reduced (albeit insignificantly in the second scenario). In any case, they may still use their knowledge about the QI to exceed the accuracy of the zero-rule baseline score by a substantial margin. From a privacy-preserving point of view, this means that a certain risk for attribute disclosure remains on the synthetic datasets of the two scenarios. Considering our customized version DSc, however, we can see that this risk is almost reduced to the baseline B. No classification algorithm performs substantially better than the zero-rule classifier, meaning that the attacker is barely able to leverage their knowledge about the QI-values of their victim. As a result, the original values of the sensitive attributes appear to be well protected against these kind of attacks.

Let us now consider the second aspect, namely the utility of the customized synthetic datasets for the main machine learning task, i.e., predicting the value of the target attributes. For this evaluation, we extended the experiments described in Section V-A to our customized versions of the DataSynthesizer for each of the two scenarios. The first rows of Tables XI and XII repeat the scores we obtained in Tables V and VI for $k = 4$. Comparing these scores to the ones for the custom Bayesian network, we observe a slight drop in the average accuracy, but no substantial loss of effectiveness. For the considered datasets and scenarios, we conclude that the customized synthetization decreases disclosure risks without causing a large effect on the utility for machine learning.

VI. Conclusions and Future Work

In this paper, we described two new techniques. The first is a genetic algorithm for speeding up the construction of Bayesian networks in data synthetization tools such as the DataSynthesizer and making them suitable for Big Data. The second is an approach for constructing Bayesian networks in a way such that the risk of disclosure of particularly sensitive attributes is reduced. We evaluated both techniques and our experiments confirm the suitability for implementing them in practical scenarios.

For future work, we will study modifications of the customized Bayesian networks, e.g. for the protection of two or more sensitive attributes or for choosing particular quasi-identifiers as those attributes that are ignored in the sampling. We will also consider the implementation of the method in other data synthetization tools such as synthpop[21] ([7]).

References

[1] L. Sweeney, "K-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, 2002.

[2] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages and Programming (ICALP)*. Venice, Italy: Springer, 2006.

[3] D. B. Rubin, Ed., *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 1987.

[4] H. Surendra and H. S. Mohan, "A review of synthetic data generation methods for privacy preserving data publishing," *International Journal of Scientific & Technology Research*, vol. 6, no. 3, 2017.

[5] H. Ping, J. Stoyanovich, and B. Howe, "Datasynthesizer: Privacy-preserving synthetic datasets," in *International Conference on Scientific and Statistical Database Management*, Chicago, IL, USA, 2017.

[6] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Montreal, QC, Canada, 2016.

[7] B. Nowok, G. Raab, and C. Dibben, "synthpop: Bespoke creation of synthetic data in R," *Journal of Statistical Software, Articles*, vol. 74, no. 11, 2016.

[8] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: Private data release via bayesian networks," *ACM Transactions on Database Systems*, vol. 42, no. 4, 2017.

[9] J. Taub, M. Elliot, M. Pampaka, and D. Smith, "Differential Correct Attribution Probability for Synthetic Data: An Exploration," in *Privacy in Statistical Databases*. Valencia, Spain: Springer International Publishing, 2018.

[10] M. Hittmeir, R. Mayer, and A. Ekelhart, "A Baseline for Attribute Disclosure Risk in Synthetic Data," in *ACM Conference on Data and Application Security and Privacy (CODASPY)*. New Orleans, LA, USA: ACM, 2020.

[11] M. Hittmeir, A. Ekelhart, and R. Mayer, "On the Utility of Synthetic Data: An Empirical Evaluation on Machine Learning Tasks," in *International Conference on Availability, Reliability and Security (ARES)*. Canterbury, United Kingdom: ACM Press, 2019.

[12] ——, "Utility and privacy assessments of synthetic data for regression tasks," in *IEEE International Conference on Big Data (Big Data)*. Los Angeles, CA, USA: IEEE, 2019.

[13] R. Mayer, M. Hittmeir, and A. Ekelhart, "Privacy-preserving anomaly detection using synthetic data," in *Data and Applications Security and Privacy (DBSec)*. Regensburg, Germany: Springer, June 2020.

[14] J. Pearl, "Bayesian networks: A model of self-activated memory for evidential reasoning," in *Proceedings of Cognitive Science Society*, 1985.

[15] D. M. Chickering, *Learning Bayesian Networks is NP-Complete*. New York, NY: Springer New York, 1996.

[16] D. Goldberg and J. Holland, "Genetic algorithms and machine learning." *Machine Learning*, vol. 3, 1988.

[17] P. Larranaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers, "Structure learning of bayesian networks by genetic algorithms: a performance analysis of control parameters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 9, 1996.

[18] J. Lee, W. Chung, and E. Kim, "Structure Learning of Bayesian Networks Using Dual Genetic Algorithm," *IEICE Transactions on Information and Systems*, vol. 91, no. 1, Jan. 2010.

[19] M. Elliot, "Final Report on the Disclosure Risk Associated with the Synthetic Data Produced by the SYLLS Team," University of Manchester, Tech. Rep., 2014. [Online]. Available: http://hummedia.manchester.ac.uk/institutes/cmist/archive-publications/reports

[20] K. El Emam, L. Mosquera, and J. Bass, "Evaluating identity disclosure risk in fully synthetic health data: Model development and validation," *Journal of Medical Internet Research*, vol. 22, no. 11, 2020.

[21] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: privacy beyond k-anonymity," in *International Conference on Data Engineering (ICDE)*. IEEE, 2006.

[21]This tool allows the user to define a so-called "predictor matrix", which for each attribute determines the set of all other attributes that are considered in the conditional sampling of its values.