Differentially Private Sum-Product Networks

Xenia Heilmann [1] **Mattia Cerrato** [1] **Ernst Althaus** [1]

Abstract

Differentially private ML approaches seek to learn models which may be publicly released while guaranteeing that the input data is kept private. One issue with this construction is that further model releases based on the same training data (e.g. for a new task) incur a further privacy budget cost. Privacy-preserving synthetic data generation is one possible solution to this conundrum. However, models trained on synthetic private data struggle to approach the performance of private, ad-hoc models. In this paper, we present a novel method based on sum-product networks that is able to perform both privacy-preserving classification and privacy-preserving data generation with a single model. To the best of our knowledge, ours is the first approach that provides both discriminative and generative capabilities to differentially private ML. We show that our approach outperforms the state of the art in terms of stability (i.e. number of training runs required for convergence) and utility of the generated data.

1. Introduction

In today's data-driven world, a vast amount of data is collected and processed each day. Data collection is not without risk, as it centralizes sensitive information of many individuals. The need for data protection is particularly present when data is shared for commercial and scientific reasons, or when models trained on these data are released to the public.

One popular method for protecting sensitive information of individuals is differential privacy (DP). It was introduced by Dwork et al. (2014) and is a formally defined privacy requirement which is one of the most robust and widely accepted privacy definitions. DP provides strong anonymization guarantees, as it formalizes that randomized computations over similar datasets have to yield similar outputs. Thus, in the event that an adversary gains access to the outputs of a DP algorithm, they cannot distinguish whether a specific datapoint is contained in the input. While DP is applicable to many use cases, it does have one limitation: Each DP model trained on the same data adds to the privacy costs and decreases the privacy guarantees for the input dataset. Therefore, training several different DP models for different tasks on the same data does not scale well in terms of privacy.

The concept of generating synthetic private data has emerged as a possible solution to this issue (Arnold & Neunhoeffer, 2020; Bowen & Snoke, 2019). If DP synthetic data is generated, it is then possible to use it to train any number of models, without incurring additional expenses of the privacy budget (due to the Post-processing Theorem – see Theorem 2.6). One issue with the current state-of-the-art methodologies is the utility-privacy tradeoff. Simply put, it is challenging to generate data that is at the same time differentially private, highly representative of the original and useful to train ML models (Tao et al., 2021). Models trained on DP synthetic data will display lower utility (e.g. accuracy) compared to DP implementations of ML models that are trained on the original non-private data.

In this paper we provide a method based on sum-product networks (Poon & Domingos, 2011) that is able to both generate private synthetic data and perform classification in a single training run. We refer to the methodology as differentially private sum-product networks (DPSPNs). These models, when trained for a classification task, may be also employed to generate DP synthetic data or perform marginal inference over the class labels, without any retraining or adaptation required. Our methodology relies on the flexibility of sum-product networks (SPNs). SPNs are rooted acyclic directed graphs representing multivariate probability distributions and their structure may in general be learned from data. Our contribution is a differentially private method to perform structure and parameter learning in SPNs. We show that DPSPNs have several advantages compared to the state-of-the-art methodologies in this space. In terms of stability, we show in Section 5.2 that our method needs a lower number of training runs to converge to an useful model com-

[1]Institute of Computer Science, Johannes Gutenberg University Mainz, Saarstraße 21, Mainz 55122, Rhineland-Palatinate, Germany. Correspondence to: Xenia Heilmann <xenia.heilmann@uni-mainz.de>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

pared to e.g. adversarial methods (Jordon et al., 2018). On top of this, our experimentation shows that the generated private data is competitive with current state-of-the-art methods in terms of privacy and utility, in both classification and density estimation tasks.

This paper is structured as follows. First, we in short introduce DP and SPNs before explaining our methology for DPSPNs. We then give a small overview over related work in the field of DP ML and private SPNs. Next, we show the performance of our DPSPN models with extensive experimental evaluation. We end with a short conclusion and an outlook on future work.

2. Preliminaries

In the following, we introduce the differential privacy (DP) concepts and properties based on Dwork et al. (2014). Furthermore, we give an introduction to SPNs.

2.1. Differential Privacy

The first step towards defining DP, is to understand the notion of *neighboring datasets*.

Definition 2.1 (Neighboring Datasets)**.** Two datsets D_1 and D_2 are **neighboring datasets** if they only differ by exactly one datapoint.

With Definition 2.1 we can now formally define DP:

Definition 2.2 (ε-Differential Privacy)**.** Let $\varepsilon > 0$ hold. Then, a mechanism $M : D \to \mathcal{R}$ is called **ε-differential private**, if for any two neighboring datasets D_1 and D_2 and for any $R \subseteq \mathcal{R}$,

$$P[M(D_1) \in R] \leq e^{\varepsilon} P[M(D_2) \in R]. \tag{1}$$

Intuitively, Definition 2.2 guarantees that a mechanism M behaves similarly on similar input datasets. More formally, the probability of seeing the same output for two neighboring datasets, can only differ by at most e^{ε}. Changing one datapoint in a dataset is therefore not likely to change the output. As relaxation to ε-DP, we have

Definition 2.3 ((ε, δ)-Differential Privacy)**.** Let $\varepsilon > 0$ and $0 < \delta \leq 1$ hold. Then, a mechanism $M : D \to \mathcal{R}$ is called (ε, δ)**-differential private**, if for any two neighboring datasets D_1 and D_2 and for any $R \subseteq \mathcal{R}$,

$$P[M(D_1) \in R] \leq e^{\varepsilon} P[M(D_2) \in R] + \delta. \tag{2}$$

Let us now move to important properties of DP, which are especially relevant during training of algorithms satisfying DP, where several iterations and therefore several accesses to the same data are needed.

Theorem 2.4 (Basic Composition)**.** *Let $M = (M_1, \ldots, M_k)$ be a sequence of ε-differentially private mechanisms, where M_i is ε_i-differential private and potentially chosen sequentially and adaptive. Then, M is $(\sum_{i=1}^{k} \varepsilon_i)$-differential private.*

This theorem applies when an algorithm accesses the same dataset k times. An improved privacy guarantee is available when assuming that a dataset is partitioned into t disjoint subsets $D^1, \ldots, D^t$, and each mechanism is applied to exactly one of these subsets (McSherry, 2009):

Theorem 2.5 (Parallel Composition)**.** *Let $i \in [0, \ldots, t]$ and let M_i be a ε-differential private mechanism which takes an arbitrary disjoint subset $D^i \subset D$ as input. Then, the composition of $M_0, \ldots, M_t$ is ε-differential private.*

One last property of DP is that applying any mapping to a DP mechanism without additional accesses to the original data preserves all DP bounds. This is especially important, as attackers cannot weaken privacy guarantees by post-processing the DP-mechanism's output. Furthermore, this theorem sets the basis for generating synthetic DP data.

Theorem 2.6 (Post-processing Theorem)**.** *Let $M : D \to \mathcal{R}$ be ε-differential private (be (ε, δ)-differential private) and let $A : \mathcal{R} \to \mathcal{S}$ be an arbitrary randomized mapping. Then, $A \circ M$ is ε-differential private (is (ε, δ)-differential private).*

2.2. Sum-Product Networks

A sum-product network (SPN) is a rooted acyclic directed graph representing a multivariate probability distribution (Poon & Domingos, 2011). An SPN consists of three different kind of nodes, namely leaf, sum and product nodes. Leaf nodes represent probability distribution functions. In a sum node, the weighted sum of the values of all child nodes is calculated, where the weights are free parameters. Product nodes compute the product of their children's output. Starting from the leaf nodes, layers of sum and product nodes ensue until a root sum node is reached.

Learning an SPN requires learning its structure, the weights leading to sum nodes as well as the leaf distribution functions. Various algorithms are available for each step, and we refer to Sánchez-Cauce et al. (2021) for a comprehensive overview. SPNs may be employed on both discrete and continuous data as well as in hybrid domains (Molina et al., 2018). After learning, probabilistic inference can be performed in time linear in the number of nodes of the network (Poon & Domingos, 2011). Here, the nodes are evaluated starting at the leaves and then propagated bottom up, so that the output is found in the root node. Additionally, several types of inference can be performed in time that is a polynomial function of the number of edges in the graph (Poon & Domingos, 2011; Sánchez-Cauce et al., 2021). These include computing marginal and posterior probabilities, ap-

proximate most probable explanation (MPE), approximate maximum a-posteriori (MAP), and approximate MAX.

SPNs can also be used for classification problems (Gens & Domingos, 2012; Peharz et al., 2019) by employing MPE on an adapted SPN structure. Another, rather new and not yet well-researched potential of SPNs is synthetic data generation (Kroes et al., 2023). Similar to other graphical models, e.g. Bayesian Networks (Zhang et al., 2017), SPNs can generate data by sampling over their learned distribution. This is done by first taking a bottom-up pass to compute marginal likelihoods and then sampling with respect to the likelihoods through a top-down pass.

3. DPSPNs

In this section, we introduce our method for training an SPN which fulfills ε-differential privacy (DPSPN). As already mentioned, SPNs are extremely versatile in the tasks they can accomplish. They support a wide range of inference tasks and are able to generate synthetic data, by sampling from their learned distribution. This makes them especially useful in settings where collected privacy-sensitive data is of interest for a wide range of different tasks. Here, we only need to train one DPSPN model to accomplish e.g. private classification and DP synthetic data generation at the same time, instead of a generator and classifier as separate models. This altogether results in better privacy guarantees as the accesses to the privacy-sensitive data are reduced. Also, no additional models trained on the same private data are released decreasing the attack possibilities.

Furthermore, DPSPNs are applicable to discrete, continuous, categorical and mixed domains. This makes them preferable over other synthetic data generators, such as MST (McKenna et al., 2021) which are restricted to either discrete or continuous data, or use a preprocessing pipeline to embed the data into a compatible domain.

3.1. Structure and Parameter Learning

Well-known SPN structure learners, such as LearnSPN (Gens & Domingos, 2013) or MSPN (Molina et al., 2018) employ a top-down approach to learn the structure and the weights of an SPN at the same time. Our approach follows the same construction, which we summarize in the following. We learn differentially private SPNs (DPSPNs) by performing the following two steps in an alternating order: *variable splitting* and *instance splitting*. These two steps respectively split the dataset vertically and horizontally, generating product and sum nodes. Variable splitting seeks to identify sets of independent columns in the dataset and induces a product node over them. Instance splitting groups rows of data into disjoint clusters, while inducing a sum node over them. A third final step, *distribution learning*, is performed when a predefined minimal number of instances η is reached. Here, a leaf node is generated and the distribution of the remaining instances is estimated, usually via a parametric distribution.

Algorithm 1 LearnDPSPN

Input: samples $\mathcal{D} = \{\mathbf{v}^m = (v_1^m, \ldots, v_N^m) | \mathbf{v}^m \sim \mathbf{V}\}_{m=1}^M$ over a set of random variables $\mathbf{V} = \{V_1, \ldots, V_N\}$; η: minimum of instances to split; ε': privacy parameter; t maximum of privacy-consuming function calls; α: threshold for binary random column splits
Output: an DPSPN $\mathcal{S}$ learned from $\mathcal{D}$
if $|\mathbf{V}| = 1$ **then**
  $\{\mathcal{D}_c\}_{c=1}^C \leftarrow \text{dp\_Kmeans}(\mathcal{D}, \varepsilon')$
  if $C > 1$ **and** $t > 1$ **then**
    $\mathcal{S} \leftarrow \sum_{c=1}^C \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \text{LearnDPSPN}(\mathcal{D}_c, \eta, \varepsilon', t-1, \alpha)$
  else
    $\mathcal{S} \leftarrow \text{learn\_dp\_histogram\_leaf}(\mathcal{D}, \varepsilon')$
  end if
else if $|\mathcal{D}| < \eta$ **or** $t = 1$ **then**
  $\mathcal{S} \leftarrow \prod_{n=1}^{|\mathbf{V}|} \text{LearnDPSPN}(\{v_n^m | v_n^m \sim V_n\}_{m=1}^M, \eta, \varepsilon', t, \alpha)$
else
  $\{\mathcal{D}_c\}_{c=1}^C \leftarrow \text{dp\_Kmeans}(\mathcal{D}, \varepsilon')$
  if $C > 1$ **and** $t > 2$ **then**
    for $\mathcal{D}_c$ **do**
      $\{\mathbf{V}_k\}_{k=1}^K \leftarrow \text{binary\_random}(\mathcal{D}_c, \alpha)$
      if $K > 1$ **then**
        $\mathcal{D}_k \leftarrow \{\mathbf{v}_k^m | \mathbf{v}_k^m \sim \mathbf{V}_k\}_{m=1}^M$
        $\mathcal{S}_c \leftarrow \prod_{k=1}^K \text{LearnDPSPN}(\mathcal{D}_k, \eta, \varepsilon', t-1, \alpha)$
      else
        $\mathcal{S}_c \leftarrow \text{LearnDPSPN}(\mathcal{D}_c, \eta, \varepsilon', t-1, \alpha)$
      end if
    end for
    $\mathcal{S} \leftarrow \sum_{c=1}^C \frac{|\mathcal{D}_c|}{|\mathcal{D}|} S_c$
  else if $C > 1$ **and** $t = 2$ **then**
    $\mathcal{S} \leftarrow \sum_{c=1}^C \frac{|\mathcal{D}_c|}{|\mathcal{D}|} \text{LearnDPSPN}(\mathcal{D}_c, \eta, \varepsilon', t-1, \alpha)$
  else
    $\mathcal{S} \leftarrow \text{LearnDPSPN}(\mathcal{D}, \eta, \varepsilon', t-1, \alpha)$
  end if
end if
RETURN $\mathcal{S}$

The general flow of this learning algorithm is not unlike well-known SPN algorithms (Poon & Domingos, 2011; Molina et al., 2018). Our approach, however, employs differentially private variants of the three aforementioned steps.

Private Instance Splitting. As described above, instance splitting in SPN learning may be thought of as a clustering step. Previous research on private clustering (Su et al., 2016) and SPN learning (Butz et al., 2018; Molina et al., 2018) has shown that the K-means algorithm is a valid choice in this setting. Thus, we employ a DP version of Lloyd's algorithm (Su et al., 2016) that bounds each dataset column's range in the interval $[-r, +r]$. After this preprocessing step, it is possible to compute the sensitivity of the algorithm by taking into account that the procedure computes count and sum queries operations on every dimension d of the dataset

for some number of iterations γ. The sensitivity of our instance splitting procedure is therefore $(dr + 1)\gamma$, and it may be made ε-differentially private via the Laplacian mechanism, i.e. by summing Laplacian noise sampled from $Lap(\frac{(dr+1)\gamma}{\varepsilon})$ to the result of each query. We set K= 2 to achieve deeper, simpler SPN structures as showed by Vergari et al. (2015). We initialize the cluster centroids following the recommendations of Su et al. (2016) which require data bounds as input. These can be defined beforehand from publicly available datasets or calculated during or before the algorithm's execution with an additional small privacy cost (Google, 2023). One well-known issue with Lloyd's algorithm is the requirement for a well-defined metric space. This assumption is broken when e.g. the dataset contains a mix of continuous and discrete variables. Here, we rely on a preprocessing solution put forward by Molina et al. (2018), who propose to both rank the feature values and apply a random linear projection followed by a non-linear function. In the transformed data, finding an approximation of the bounds for centroid initialization comes to no additional privacy cost. We elaborate on these details in Appendix A.3.

After the clustering step is completed, instance splitting still requires to induce a sum node while learning the parameters of the connection between the sum node and its children. Here, we straightforwardly set the weights as the size of a cluster divided by the size of data samples given to the K-means algorithm as input. Since this may be done with no additional accesses to the dataset, Theorem 2.6 applies and no further privacy budget is consumed.

Private Variable Splitting. Variable splitting requires finding sets of independent variables/columns in the dataset. After some investigation, we found that differentially private versions for independency tests such as the G-test (Gens & Domingos, 2013) require ample privacy budget without providing much improvement in terms of resulting model quality. One further observation that we gather from prior literature on SPNs (Peharz et al., 2019) is that random structures are often competitive with optimal or near-optimal structures. Thus, we chose to implement a binary random variable splitting method as the basis for DPSPNs. This method randomly partitions the variables into two random subsets with a failure percentage of α. The failure percentage denotes with which probability the column splitting method will not split the variables into subsets but fail. This imitates the behavior of independency tests, as these can also fail if no independence is found between the remaining variables. The size of both variable partitions is then drawn from a Beta distribution (Molina et al., 2019). This leads to structures that are similar to the ones generated by LearnSPN. We stress that our variable splitting procedure is data-independent, and therefore comes at no cost of the privacy budget. We show the performance of this methodology compared esp. with avoiding variable splits at all in Figure 2.

Algorithm 2 LearnDPSPN_class

Input: samples $\mathcal{D} = \{\mathbf{v}^m = (v_1^m, \ldots, v_N^m) | \mathbf{v}^m \sim \mathbf{V}\}_{m=1}^M$ over a set of random variables $\mathbf{V} = \{V_1, \ldots, V_N\}$; η: minimum of instances to split; ε': privacy parameter; l: index of the label; t maximum of privacy-consuming function calls, α: threshold for binary random column splits
Output: an DPSPN $\mathcal{S}$ learned from $\mathcal{D}$ which is able to classify data
$\mathcal{L} \leftarrow \{v_l^m | v_l^m \sim V_l\}_{m=1}^M$
for i in unique($\mathcal{L}$) **do**
  $\mathcal{D}_i \leftarrow \{\mathbf{v} \in \mathcal{D} | v_l = i\}$
  $\mathcal{S}_i \leftarrow$ LearnDPSPN($\mathcal{D}_i, \eta, \varepsilon', t-1, \alpha$)
  $w_i \leftarrow |\{v_l^m \in \mathcal{L} | v_l^m = i\}| + Z$ with $Z \sim \frac{1-e^{-\varepsilon'}}{1+e^{-\varepsilon'}} e^{-\varepsilon'|z|}$
end for
RETURN $\mathcal{S} \leftarrow \sum_i \frac{w_i}{\sum_j w_j} \mathcal{S}_i$

Private Distribution Learning. In general, leaf nodes are generated once a minimal number of instances η is reached. Here, the challenge was to find a distribution learning method which can handle data from categorical, discrete and continuous datasets. One such approach is to build Laplacian-smoothed DP histograms of the data (Dwork et al., 2014; Molina et al., 2018). Concretely, this means that we add two-sided geometric distributed noise $Z \sim \frac{1-e^{-\varepsilon}}{1+e^{-\varepsilon}} e^{-\varepsilon|z|}$ to the counting function (which has sensitivity 1) to ensure DP. To cope with potentially unseen values in the data, we employ Laplacian smoothing.

Enabling Classification. As already mentioned, SPNs can also be used for classification tasks. For DPSPNs, this can be realised with only slight modifications to Algorithm 1. In the root level we introduce a sum node with individual DPSPNs as children where each DPSPN represents one classification class in the data. The weight of the edges to the sum node are then calculated as the class counts divided by the size of the whole data, adding two-sided geometric distributed noise (as for private distribution learning) to the class counts to preserve differential privacy. Concretely, it is that the the new root sum weights represent the class probabilities and each sub-SPNs class conditional distributions. At inference time, approximate most probable explanation (MPE) is applied to find the class value. Here, one path from leaves to root is needed to calculate the maximal configurations for each node. Then, the algorithm backtracks to find the class value which maximizes the probability of the input. The modified method can be found in Algorithm 2.

Analysis of Privacy Costs. In the following, we summarize the general structure of the DPSPN learning algorithm (Algorithm 1) so to better understand the total privacy costs involved in learning. After generating the root node, the dataset is split into disjunct subsets which are fed into child nodes via alternating, recursive calls of the variable splitting

and instance splitting functions. Therefore, in each level of the DPSPN we only access the whole dataset once. As all functions called on the data are ε'-DP, we can then claim that the resulting privacy guarantee of a DPSPN is $\varepsilon' \cdot layers$ by exploiting Theorem 2.4. However, we need to also take into account that during the process of finding the next node several functions may be called. In the worst case, structure learners can cycle between trying to find variable and instance splits. A failure can happen as for variable splits we have a failure percentage of α and for instance splits only one cluster could be returned. To control the privacy guarantee and prevent extensive accesses to the same subset of data, we have to limit this cycling. Therefore, we introduce an upper bound t for the calls to privacy-consuming functions (i.e. instance splitting and distribution learning). In practice, we force the generation of leaf nodes after $t-1$ function calls. We then have a maximal privacy guarantee independent of the learned DPSPN, that can be calculated before the training process starts. We now can state the following theorem:

Theorem 3.1. *Given a dataset $\mathcal{D}$, t limiting the total privacy-consuming operations on $\mathcal{D}$ and $\varepsilon' > 0$ as input, DPSPNs, as proposed in Algorithm 1 are ε-differential private, with $\varepsilon = \varepsilon' \cdot t$.*

For learning DPSPNs which are able to classify data, we have a similar theorem based on Algorithm 2:

Theorem 3.2. *Given a dataset $\mathcal{D}$, t limiting the total privacy-consuming operations on $\mathcal{D}$ and $\varepsilon' > 0$ as input, DPSPNs for classification, as proposed in Algorithm 2 are ε-differential private, with $\varepsilon = \varepsilon' \cdot t$.*

The formal proofs can be found in Appendix A.1 and A.2. Here, note that in practice it can happen that t is not reached before the algorithm ends which makes Theorem 3.1 and 3.2 an upper bound to the privacy costs.

DPSPNs can be learned on a wide range of datasets and applied to a a variety of tasks. Thanks to the Post-processing Theorem (Theorem 2.6), one can apply different inference algorithms or generate DP synthetic data from a trained DPSPN without incurring in any additional privacy loss.

4. Related Work

In this section we shortly introduce related work on DP classification algorithms, DP synthetic data generation and private SPNs.

With regard to classification, centralized non-private ML models are often optimized with variants of stochastic gradient decent (SGD), esp. those based on neural networks. DP versions of these algorithms employ a noisy SGD variant (Abadi et al., 2016). Other classification models have also been adapted to DP settings, e.g. Naïve Bayes for classification (Vaidya et al., 2013). In distributed settings where privacy is relevant, the PATE framework (Papernot et al., 2016) and its extensions (Jordon et al., 2018) have proven influential. These methods are based on the teacher-student framework and exploit parallel composition (see Theorem 2.5) to enable strong privacy guarantees.

DP synthetic data has become a popular research area at least since the National Institute of Standards and Technology Public Safety Communication Research (NIST PSCR) Division's "Differential Privacy Synthetic Data Challenge" (NIST, 2018) in 2018. There exist many algorithms tackling DP synthetic data generation, ranging from Generative Adversarial Networks (Jordon et al., 2018) to Bayesian Networks (Zhang et al., 2017). The challenge is to minimize privacy costs while at the same time preserving the utility of the generated data. While many models exist which can do either DP inference or DP synthetic data generation, there is, to the best of our knowledge, no work on DP models which are able to generate data and train a classifier at the same time. Currently, to enable both, two different DP models trained on the same data would be of need. Our approach is instead to learn a joint classification and data generation SPN model, which only needs to be trained once.

To the best of our knowledge, previous work on privacy-preserving SPNs is limited to the private inference protocol CRYPTOSPN (Treiber et al., 2020). This method is based on Yao's garbled circuit (Yao, 1982) and implements a two-party cryptographically secure inference protocol between a client and a server. Thus, the client learns nothing about the model beyond its outputs, while the server never holds the client's data in cleartext. CryptoSPN is a fast inference method, but is only implemented as a two-party protocol. In our work, we learn differentially private SPNs that may thus be shared publicly with any party, enabling both private inference and private synthetic data generation.

5. Experimental Evaluation

To show the potential of our proposed method, we conducted extensive experiments on classification, density estimation and DP synthetic data generation tasks.[1] In this section we intend to discuss the following research questions:

(Q1) How stable are DPSPNs?

(Q2) How does ε influence the classification performance of DPSPNs?

(Q3) How does DPSPN-generated synthetic data compare to existing DP methods for synthetic data generation?

(Q4) How do DPSPNs perform in density estimation?

[1] Code for all the experiments is available at https://github.com/xheilmann/DPSPN.

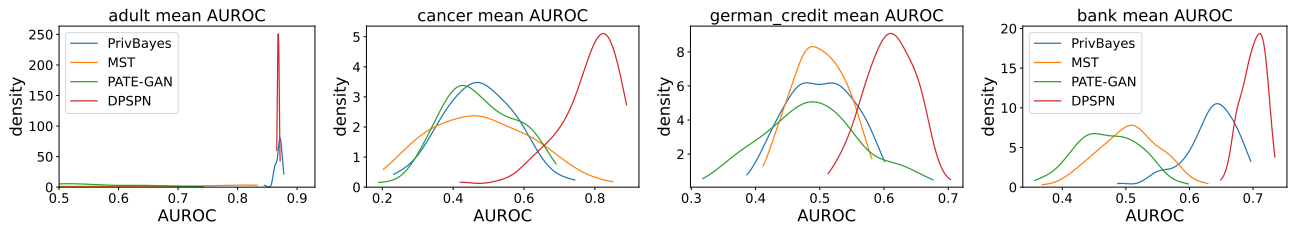

Figure 1. Distribution of test AUROC over 50 training runs for four classification dataset, mean over 5 models trained on DP synthetic data.

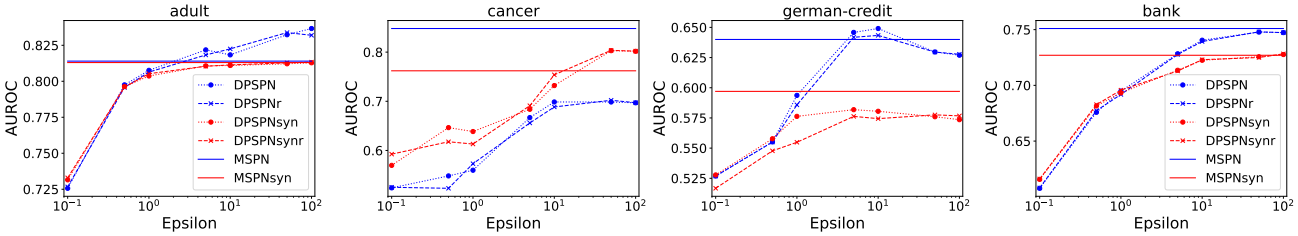

Figure 2. Comparison of AUROC performance for DPSPNs and generated synthetic data by DPSPNs (on which an MSPN was trained afterwards) with and without binary random column splitting and different epsilon values. AUROC performance for MSPNs and generated synthetic data by MPSPNs is given as baseline.

5.1. Experimental Setup

For both inference and generative tasks we use the same learned DPSPN, so to showcase their flexibility. In order to differentiate between the two tasks, however, we introduce the following notation: We use DPSPNsyn to denote the synthetically generated data by an DPSPN, which is then used to train a predictive model or a non-private MSPN. For classification tasks, we provide results for five datasets with continuous, discrete and binary variables: cervical cancer (mis, 2019), german-credit (Hofmann, 1994), diabetes (Semerdjian & Frank, 2017), bank (Moro et al., 2014) and adult (Becker & Kohavi, 1996) (see Appendix B, Table 3 for dataset statistics). We compare the performance in terms of area under the receiver operating characteristic (AUROC) and area under the precision-recall curve (AUPRC) of the DPSPNs to MSPNs which use K-means for row splitting, binary random column splits and histograms as leaves. Here, datasets from hybrid domains are preprocessed by the pipeline described in Appendix A.3. For MSPNs, we applied random hyperparameter search to optimize the binary splitting criteria α and minimal instance parameter η. For DPSPNs, we performed a grid search over $\eta = \{0.1N, 0.2N, \ldots, N\}$ (with N defining the size of the dataset) and $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$ to give an intuition to which hyperparameters to choose for unknown datasets. Furthermore, we report the results with and without binary random column splits. We also note that we optimized DPSPN hyperparameters with the classification AUROC score as an objective. Then, we reported the same model's performance in synthetic data generation.

To evaluate how well DPSPNs generate DP synthetic data, we compared our method to PATE-GAN (Jordon et al., 2018), MST (McKenna et al., 2021) and PrivBayes (Zhang et al., 2017). The latter two we chose due to their participation in the NST PCR "Differential Privacy Synthetic Data Challenge" (NIST, 2018) and the therefore resulting visibility. However, there are other DP synthetic data generators available, e.g. AIM (McKenna et al., 2022), P3GM (Takagi et al., 2022) or GS-WGAN (Chen et al., 2020). The MST method won the NST PCR "Differential Privacy Synthetic Data Challenge" (NIST, 2018) contest. MST uses graphical models based on sanitized marginals to model the data distribution. For MST, we apply predefined parameters as proposed by McKenna et al. (2021). However, the MST method is restricted to generating discrete-valued data. We did report AUROC and AUPRC scores for all five datasets for this method, however bank and adult are the sole datasets with only discrete values.

The PATE-GAN method is based on Generative Adversarial Nets (GAN) and the Private Aggregation of Teacher Ensembles (PATE) framework (Papernot et al., 2016). PATE-GAN normalizes the training set into $[0, 1]^d$ before training starts. To compare it with DPSPNs we therefore mapped the generated data by PATE-GAN back to the original space, rounding binary and discrete attributes to integers. We also

Table 1. Mean performance on real test data of 5 different predictive models trained on synthetic DP data generated by 4 different methods with varying ε. The performance of the original dataset is given as comparison.

		AUROC					AUPRC				
ε	dataset	Orig.	DPSPN	MST	PATE-GAN	PrivBayes	Orig.	DPSPN	MST	PATE-GAN	PrivBayes
0.1	adult	0.89	**0.86**	0.77	0.49	0.78	0.75	**0.66**	0.45	0.23	0.48
	cancer	0.85	0.7	**0.73**	0.54	0.53	0.48	0.2	**0.3**	0.08	0.12
	diabetes	0.76	**0.61**	0.51	**0.61**	0.52	0.39	**0.28**	0.24	**0.28**	0.25
	german-credit	0.65	0.54	0.52	**0.58**	0.55	0.4	0.31	0.31	**0.32**	0.31
	bank	0.86	**0.65**	0.55	0.54	0.64	0.87	**0.65**	0.55	0.54	0.63
1	adult	0.89	0.87	0.69	0.73	**0.88**	0.75	0.69	0.51	0.49	**0.71**
	cancer	0.85	**0.83**	0.82	0.56	0.72	0.48	**0.46**	0.41	0.13	0.22
	diabetes	0.76	**0.66**	0.51	0.63	0.59	0.39	**0.38**	0.25	0.34	0.28
	german-credit	0.65	**0.65**	0.55	0.64	0.59	0.4	**0.4**	0.33	0.39	0.37
	bank	0.86	**0.71**	0.56	0.57	**0.71**	0.87	**0.7**	0.55	0.58	**0.7**
10	adult	0.89	**0.88**	0.84	0.66	**0.88**	0.75	0.7	0.65	0.34	**0.74**
	cancer	0.85	**0.89**	0.78	0.53	0.84	0.48	**0.63**	0.42	0.11	0.33
	diabetes	0.76	**0.66**	0.51	0.6	0.63	0.39	**0.38**	0.25	0.29	0.31
	german-credit	0.65	**0.67**	0.53	0.57	0.59	0.4	**0.42**	0.3	0.35	0.35
	bank	0.86	**0.76**	0.59	0.57	0.7	0.87	**0.77**	0.59	0.56	0.7

optimized the number of teachers by choosing it in the set $\{N/10, N/50, N/100, N/500, N/1000, N/5000, N/10000\}$ as given in the original paper by Jordon et al. (2018).

PrivBayes (Zhang et al., 2017) is another competitive method for synthetic data generation that was included in the NST PCR "Differential Privacy Synthetic Data Challenge" contest. The method is based on a Bayesian network. For PrivBayes we used $\beta = 0.3$ and $\theta = 4$ as input parameters and the vanilla encoding (Zhang et al., 2017, Section 5.1) to encode the input datasets.

All the methods described above are $(\varepsilon, \delta)-$differential private. DPSPNs however fulfill the stronger ε-differential privacy. We conducted experiments for $\varepsilon \in \{0.1, 0.5, 1, 5, 10, 100\}$ and kept a fixed $\delta = 10^{-6}$. Additionally, for DPSPNs we set the maximum of privacy-consuming function calls on the data t to 10.

5.2. Experimental Results

How stable are DPSPNs? DP methodologies often suffer from notable variance in performance from one full training cycle (training run) to another. The main reason for this behavior is the noisiness of the initialization and training process, which will necessarily include some form of randomization. One possible solution is to start multiple training runs and only release the best-performing model. Nonetheless, we argue that reducing the performance variance, in other words improving the *stability*, of an DP methodology is desirable. Firstly, we reason that the discarded models may still be obtained by untrusted parties. Furthermore, the party responsible for model training may not be a trusted party itself, only accessing the data via e.g. a private model training API which does not provide the data in cleartext, only returning the resulting model. In these scenarios, each training run subtracts from the privacy budget and a more stable model will be clearly more desirable. We show the stability of DPSPNs by plotting the distribution of mean AUROC scores over five predictive models trained on DPSPN-generated data. The results are taken from 50 DPSPN training runs and may be found in Figure 1 (results for the diabetes dataset can be found in the Appendix Figure 6). We compare this distribution with ones obtained by employing MST, PrivBayes and PATE-GAN in the same experimental conditions. Here, one can see that DPSPNs throughout all 50 training runs show a very stable performance on all 5 datasets, with limited variance and a higher expected performance. PrivBayes also has a remarkably stable performance on datasets which have only discrete variables (bank and adult). We observe that PrivBayes outperforms DPSPNs in some runs on the adult dataset, not showing the constant stable performance of DPSPNs. Altogether, we conclude that training a single DPSPN is sufficient to achieve good overall results.

Another facet of stable performance is a method's performance over different hyperparameter settings. A solution here is to optimize hyperparameters on a publicly available dataset, or obtain good suggestions on how to choose hyperparameters for unknown datasets. We report the performance of various hyperparameter configurations for classification DPSPNs in the Appendix, Figure 4 due to space limitations. We observe that the choice of hyperparameters is not critical, and performance stays stable across different values. For datasets on which density estimation is performed, we observe however that the minimal number of instances η should be in the interval $\{0.4N, \ldots, 0.9N\}$ to yield good performance (see Appendix, Figure 5).

How does ε influence the classification performance of DPSPNs? In Figure 2 we can see the trade-off between

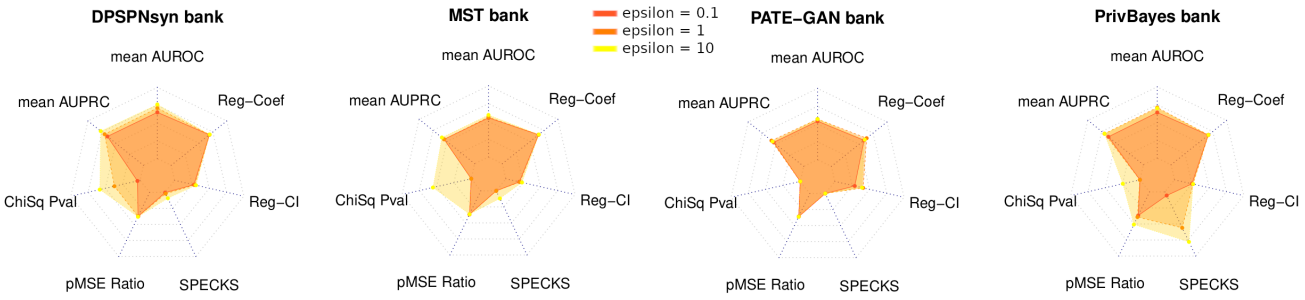

Figure 3. Utility summary charts for the bank dataset.

AUROC performance and privacy guarantee for DPSPNs with and without binary random column splits (an *r* is added to the method's name for binary random column splits). We show results for prediction tasks as well as DPSPN-generated synthetic data on which an MSPN is then trained. As a non-private baseline, we also include the performance of MSPNs and synthetic data generated by MSPNs on which an MSPN is trained afterwards. DPSPNs show a superior performance over MSPNs trained on synthetic data generated by DPSPNs for all datasets except cancer. This validates that combining the discriminative and generative abilities of DPSPNs improves the accuracy at inference time and at the same time provides the flexibility to generate DP synthetic data for application to other models (see **Q3**).

Furthermore, we can see that as ε increases, the AUROC performance of both DPSPNs and their generated data reaches the non-private baseline. It is however relevant to mention that prior work has established that reasonable values for ε are < 10, while values < 1 are preferable (Jayaraman & Evans, 2019). Our results show that for $\varepsilon > 1$ AUROC performance on some datasets has already reached or slightly surpassed the baseline, suggesting that the privacy costs are leveled with the benefit of injecting a certain amount of noise into the models. Interestingly, binary random column splits only improve the AUROC performance for the synthetically generated data for diabetes and german-credit. This suggests that the choice of a variable splitting method is of less importance than the choice of an instance splitting mechanisms.

How does DPSPN-generated synthetic data compare to existing DP methods for synthetic data generation? To analyse how well our method compares to other DP generative methods, we generated synthetic data with DPSPN, MST, PATE-GAN and PrivBayes. Here, to handle initialization randomness, the best model over 10 training runs was used for data generation based on the evaluation methodology of PATE-GAN (Jordon et al., 2021). However, we stress that DPSPNs are not prone to initialization randomness and we only adopt this methodology for a more robust comparison. The data was then used to train 5 different predictive models – Logistic Regression, Random Forest, Neural Network, Gaussian Naïve Bayes, Gradient Boosting Classifier. We report the mean AUROC and AUPRC performance of all five models in Table 1. The best results are in bold, showing an overall superior performance of DPSPNs over the other methods. We also follow the recommendation for holistic evaluation of synthetic generated data put forward by Bowen & Snoke (2019) and Arnold & Neunhoeffer (2020) among others. Beyond predictive performance on a classification tasks, these contributions suggest to evaluate the generated data in terms of fidelity to the original distribution. To this end, various fidelity metrics may be computed, comparing either the marginal distributions (via the χ^2 and KS tests), the joint (pMSE ratio, SPECKS) or even the correlation between different dataset columns (confidence interval overlap of a regressor). A detailed introduction to these metrics can be found in Appendix D. We follow the same evaluation and visualization protocol introduced by Bowen & Snoke (2019), reporting in Figure 3 the summary chart of DPSPNs on the bank dataset. Further summary charts are available in the Appendix (Figures 7 to 10). We observe that DPSPNs outperform other methods in terms of similarity of the marginal distributions of the new variables when $\varepsilon = 0.1$. However, other models based on graphical models such as PrivBayes have greater joint distribution fidelity at higher ε values.

How do DPSPNs perform in density estimation? We conducted experiments on a set of binary datasets (see Appendix B Table 4 for dataset statistics) commonly used to evaluate the performance of SPNs (Gens & Domingos, 2013; Peharz et al., 2019). Our purpose here is to investigate the gap between our DPSPNs and regular SPNs in a well-established evaluation setting for these models. We applied the MSPN algorithm with k-means for instance splitting to learn the non-private SPNs. A comparison to other density estimation models as well as different learning algorithms for SPNs can be found in Peharz et al. (2019, Table 1). In Table 2, we see that for most datasets the log-likelihood performance of the DPSPN reaches the SPN baseline as ε increases. We report further results in Appendix E Table 11, in which one can observe the same trends.

Table 2. SPN and DPSPN log-likelihood performance for the binary datasets and different epsilon values, average of 10 runs.

dataset	SPN	DPSPN		
		$\varepsilon = 0.1$	$\varepsilon = 1$	$\varepsilon = 10$
kdd	-2.58	-2.72	-2.63	-2.61
nltcs	-6.0	-6.93	-6.53	-6.4
tretail	-11.13	-11.93	-11.37	-11.3
kosarek	-11.66	-14.01	-12.62	-12.4
accidents	-37.32	-44.87	-43.1	-42.26
pumsb_star	-38.75	-57.66	-54.64	-44.11
bnetflix	-58.93	-62.31	-60.67	-59.23
tmovie	-61.23	-95.59	-78.27	-76.25
c20ng	-124.2	-139.8	-132.43	-131.86
cwebkb	-160.68	-212.56	-179.23	-171.81

6. Future Work

Overall, we have shown that DPSPNs are very stable models. They are versatile in the tasks they can accomplish and can generate DP synthetic data comparable to existing state-of-the-art methods in terms of utility-privacy trade-off. While we have mainly focused on classification and data generation, there are still many tasks left unexplored, in particular those requiring approximation of higher-dimensional distributions. As DPSPNs are learned in an iterative way, further research could analyse an uneven distribution of the privacy budget between the different functions or iteration steps throughout the learning algorithm. Here, as the iteration steps resemble the layers of the DPSPN, one idea could be to allocate more privacy budget towards the lower layers for better representation of the data, allowing for more noisiness and approximation in the top layers. Another venue for further research is to extend our current method to approximate DP, i.e. (ε, δ)-DP.

Acknowledgements

The research in this paper was funded by the Carl Zeiss Foundation, grant number P2021-02-014 (TOPML project).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Cervical Cancer Behavior Risk. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5402W.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Arnold, C. and Neunhoeffer, M. Really useful synthetic data–a framework to evaluate the quality of differentially private synthetic data. *arXiv preprint arXiv:2004.07740*, 2020.

Becker, B. and Kohavi, R. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Bowen, C. M. and Snoke, J. Comparative study of differentially private synthetic data algorithms from the nist pscr differential privacy synthetic data challenge. *arXiv preprint arXiv:1911.12704*, 2019.

Butz, C. J., Oliveira, J. S., Santos, A. E., Teixeira, A. L., Poupart, P., and Kalra, A. An empirical study of methods for spn learning and inference. In *International Conference on Probabilistic Graphical Models*, pp. 49–60. PMLR, 2018.

Chen, D., Orekondy, T., and Fritz, M. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. *Advances in Neural Information Processing Systems*, 33:12673–12684, 2020.

Dwork, C., Roth, A., et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Gens, R. and Domingos, P. Discriminative learning of sum-product networks. *Advances in Neural Information Processing Systems*, 25, 2012.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. In *International conference on machine learning*, pp. 873–880. PMLR, 2013.

Ghosh, A., Roughgarden, T., and Sundararajan, M. Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 351–360, 2009.

Google. differential-privacy, 04 2023. URL https://github.com/google/differential-privacy.

Hofmann, H. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

Jayaraman, B. and Evans, D. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1895–1912, 2019.

Jordon, J., Yoon, J., and Van Der Schaar, M. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International conference on learning representations*, 2018.

Jordon, J., Yoon, J., and Van Der Schaar, M. Codebase for "pate-gan: Generating synthetic data with differential privacy guarantees", 2021. URL https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/pategan.

Kroes, S. K., van Leeuwen, M., Groenwold, R. H., and Janssen, M. P. Generating synthetic mixed discrete-continuous health records with mixed sum-product networks. *Journal of the American Medical Informatics Association*, 30(1):16–25, 2023.

McKenna, R., Miklau, G., and Sheldon, D. Winning the nist contest: A scalable and general approach to differentially private synthetic data. *arXiv preprint arXiv:2108.04978*, 2021.

McKenna, R., Mullins, B., Sheldon, D., and Miklau, G. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677*, 2022.

McSherry, F. D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 19–30, 2009.

Molina, A., Vergari, A., Di Mauro, N., Natarajan, S., Esposito, F., and Kersting, K. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Molina, A., Vergari, A., Stelzner, K., Peharz, R., Subramani, P., Mauro, N. D., Poupart, P., and Kersting, K. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks. *CoRR*, abs/1901.03704, 2019. URL http://arxiv.org/abs/1901.03704.

Moro, S., Cortez, P., and Rita, P. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

NIST. "differential privacy synthetic data challenge", 2018. https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2018-differential-privacy-synthetic [Accessed: (19.12.2023)].

Papernot, N., Abadi, M., Erlingsson, U., Goodfellow, I., and Talwar, K. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Shao, X., Kersting, K., and Ghahramani, Z. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In Globerson, A. and Silva, R. (eds.), *Proc. of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, volume 115 of *Proceedings of Machine Learning Research*, pp. 334–344. AUAI Press, 2019. URL http://proceedings.mlr.press/v115/peharz20a.html.

Póczos, B., Ghahramani, Z., and Schneider, J. Copula-based kernel dependency measures. *arXiv preprint arXiv:1206.4682*, 2012.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 689–690. IEEE, 2011.

Sánchez-Cauce, R., París, I., and Díez, F. J. Sum-product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Semerdjian, J. and Frank, S. An ensemble classifier for predicting the onset of type ii diabetes. *arXiv preprint arXiv:1708.07480*, 2017.

Su, D., Cao, J., Li, N., Bertino, E., and Jin, H. Differentially private k-means clustering. In *Proceedings of the sixth ACM conference on data and application security and privacy*, pp. 26–37, 2016.

Takagi, S., Takahashi, T., Cao, Y., and Yoshikawa, M. P3gm: Private high-dimensional data release via privacy preserving phased generative model, 2022.

Tao, Y., McKenna, R., Hay, M., Machanavajjhala, A., and Miklau, G. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238*, 2021.

Treiber, A., Molina, A., Weinert, C., Schneider, T., and Kersting, K. Cryptospn: Expanding ppml beyond neural networks. In *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, pp. 9–14, 2020.

Vaidya, J., Shafiq, B., Basu, A., and Hong, Y. Differentially private naive bayes classification. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pp. 571–576. IEEE, 2013.

Vergari, A., Di Mauro, N., and Esposito, F. Simplifying, regularizing and strengthening sum-product network structure learning. In *Machine Learning and Knowledge Discovery in Databases*, pp. 343–358. Springer International Publishing, 2015. ISBN 978-3-319-23525-7.

Yao, A. C. Protocols for Secure Computations. In *Proc. of the 23rd Annual Symposium on Foundations of Computer Science*, pp. 160–164. IEEE, 1982.

Zhang, J., Cormode, G., Procopiuc, C. M., Srivastava, D., and Xiao, X. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):1–41, 2017.

A. Proofs.

A.1. Proof of Theorem 3.1.

Theorem. *Given a dataset $\mathcal{D}$, t limiting the total privacy-consuming operations on $\mathcal{D}$ and $\varepsilon' > 0$ as input, DPSPNs, as proposed in Algorithm 1 are ε-differential private, with $\varepsilon = \varepsilon' \cdot t$.*

Proof. We know that K-means and histograms with input ε are ε-DP (Su et al., 2016; Dwork et al., 2014). Furthermore, the binary random column splits do not increase the privacy costs. Left to show is that the overall privacy costs do not exceed $\varepsilon = \varepsilon' \cdot t$ with t limiting the total privacy-consuming operations on the data. We have the following possible operations: K-means clustering (potentially followed by binary splitting) and generation of a histogram leaf. Now, to calculate the final ε-costs we have to count how many times each operation is performed on the whole dataset. We know, that the leaf function is only called once for each datapoint. For K-means we have to take into account that the top down learning approach starts with the whole dataset generating the root node, followed by dataset splits which split the data into disjunct subsets. Therefore, we have that in each layer of nodes, each node n_j is generated from a disjunct subset of the data which makes Theorem 2.5 applicable. Here, we also have to keep in mind that not every call to K-means returns more than one cluster. It can happen, due to a bad initialization of the centroids that two or more clusters are only found after several calls of the K-means algorithm.

Let us now concretely look at one child n_0 of the root note. Let us denote the number of calls for generation of n_0 with k_0. Then, with t given, we know that $k_0 < t$ has to hold (otherwise we directly generate a leaf node layer). With Theorem 2.4, this gives us a privacy cost of $k_0 \cdot \varepsilon'$ for n_0. Now, to generate a child node n_1 of n_0, we access subsets of the data on which already k_0 DP operations where performed. Thus, we now can only perform another $k_1 < (t - k_0)$ DP operations. By induction, we have that for node n_i the privacy costs are $(k_0 + \cdots + k_i) \cdot \varepsilon' < t \cdot \varepsilon'$. When $(k_0 + \cdots + k_i) \cdot \varepsilon' = (t-1) \cdot \varepsilon'$ holds, leaves are generated with one additional access to the data so that altogether the privacy cost add up to $\varepsilon' \cdot t = \varepsilon$. With Theorem 2.5, this holds for all children of the root node giving us the upper bound for the privacy costs.

□

A.2. Proof of Theorem 3.2.

Theorem. *Given a dataset $\mathcal{D}$, t limiting the total privacy-consuming operations on $\mathcal{D}$ and $\varepsilon' > 0$ as input, DPSPNs for classification, as proposed in Algorithm 2 are ε-differential private, with $\varepsilon = \varepsilon' \cdot t$.*

Proof. Let L denote the number of unique labels v_l and $\mathcal{D}$ and $\mathcal{D}'$ two neighboring datasets. We have that all $\mathcal{D}_i = \{\mathbf{v} \in \mathcal{D} | v_l = i\}$ with $i \in \{1, \ldots, L\}$ are disjoint by definition. Then, we know that the $\mathrm{LearnDPSPN\_class}()$ maximal differs for one $\mathcal{D}_j \subseteq \mathcal{D}$ and $\mathcal{D}_{j'} \subseteq \mathcal{D}'$. We already know with Theorem 3.1 that $\mathrm{LearnDPSPN}()$ with input ε' and $t-1$ as data operation limit is $\varepsilon' \cdot (t-1)$-DP. We also know that adding noise sampled from the two-sided geometric distribution $Z \sim \frac{1-e^{-\varepsilon'}}{1+e^{-\varepsilon'}} e^{-\varepsilon'|z|}$ to count functions is ε'-DP. Together with Theorem 2.4, we have that the operations in the loop fulfill $\varepsilon' \cdot (t-1+1) = \varepsilon$-DP (Ghosh et al., 2009). With Theorem 2.5, we then have that the whole loop satisfies ε'-DP. With Theorem 2.6 we have that the last division does not increase the privacy-budget, so that altogether $\mathrm{LearnDPSPN\_class}()$ is ε'-DP. □

A.3. Transformation of Hybrid Data.

Furthermore, we want to discuss why the choice of the data bounds after the transformation pipeline for data from hybrid domains (Molina et al., 2018) with N datapoints does not decrease the privacy guarantees and is valid. We first introduce the pipeline which proceeds in the following steps:

1. At first the empirical compula tranformations (Póczos et al., 2012) is performed for each variable in the set of variables $\mathbf{V} = \{V_1, \ldots, V_I\}$ in the data:

$$\mathcal{C}_{V_i} \leftarrow \left\{ \frac{1}{N} \sum_{r=1}^{N} \mathbb{1}\{v_i^r \leq v_i^n\} | v_i^n \in \mathcal{D}_{V_i} \right\}_{n=1}^{N}$$

   Informally, this steps ranks the values of one variable by their value, while same values always get assigned the same and highest possible rank. By division through N we have that $\forall c_i \in \mathcal{C}_{V_i} : 0 < c_i \leq 1$ and 1 is always in $\mathcal{C}_{V_i}$.

2. Then, a random linear projection into a k-dimensional space is performed, followed by a non-linear function (here sinus)

$$\phi(\mathcal{C}_{V_i}) \leftarrow \sin(\mathbf{w}\mathcal{C}_{V_i}^T + b), \text{ with } (\mathbf{w}, b) \sim \mathcal{N}(\mathbf{0}_k, s\mathbf{I}_{k\times k}).$$

We choose $s = \frac{1}{6}$ as proposed by Molina et al. (2018). Then, it follows as $\mathcal{C}_{V_i}^T$ is fixed, that

$$\mathbf{w}\mathcal{C}_{V_i}^T + b \sim \mathcal{N}(\mathbf{0}_k, (\frac{1}{6}\mathcal{C}_{V_i}^T + \frac{1}{6})\mathbf{I}_{k\times k}).$$

For bound calculation without privacy loss we want a good approximation of the maximal and minimal value which $\sin(\mathbf{w}\mathcal{C}_{V_i}^T + b)$ can take on over all V_i. As said above we have that the values of $\mathcal{C}_{V_i}^T$ lie in the interval $(0, 1]$, so that $\mathbf{w}\mathcal{C}_{V_i}^T + b$ is upper and lower bounded by the distribution $\mathbf{w}\mathbb{1}_k + b \sim \mathcal{N}(\mathbf{0}_k, (\frac{1}{6} + \frac{1}{6})\mathbf{I}_{k\times k})$. It holds that 68% of the values drawn from this normal distribution are in the interval $[-\frac{1}{\sqrt{3}}, +\frac{1}{\sqrt{3}}]$. With the non-linearity of the sinus function we then have that 68% of values lie in $[\sin(-\frac{1}{\sqrt{3}}), \sin(\frac{1}{\sqrt{3}})]$. This gives us as bound approximation -0.5458 as lower and 0.5458 as upper bound. One can argue that these bounds are too tight as we only have 68% of the values in this interval. Yet, it is seldom true that all $v_i \in V_i$ take on the same values (which would correspond in all entries of $\mathcal{C}_{V_i}^T$ being 1), so that one can expect to have more than 68% of the values in the given interval.

B. Dataset Statistics.

In Table 3 and 4 the dataset statistics for the evaluated datasets are listed.

Table 3. Dataset statistics for the classification datasets.

dataset	#datapoints	discrete/binary	continuous
adult	36632	14	0
diabetes	1723	1	16
cancer	502	29	5
german-credit	720	20	1
bank	8372	17	0

Table 4. Dataset statistics for the binary datasets.

dataset	#variables	#training set	density
kdd	65	180092	0.008
nltcs	16	16181	0.332
tretail	135	22041	0.024
kosarek	190	33375	0.020
accidents	111	12758	0.291
pumsb_star	163	12262	0.270
bnetflix	100	15000	0.541
tmovie	500	4524	0.059
c20ng	910	11293	0.049
cwebkb	839	2803	0.064
msnbc	17	291326	0.166
msweb	294	29441	0.010
plants	69	17412	0.180
book	500	8700	0.016
baudio	100	15000	0.199
ad	1556	2461	0.008
jester	100	9000	0.608
cr52	889	6532	0.036
dna	180	1600	0.253
bbc	1058	1670	0.078

C. Choice of Hyperparameters.

In Figure 4 and 5 we show the development of the AUROC score of DPSPNs with regard to the parameter η defining the minimum of instances at which no further data splits are performed. Once the minimum of instances is reached, leaf nodes

are generated. We choose η as a percentage of the dataset size N, with $0.1N$ as lowest and N as highest value, performing 10 runs with each configuration. For the classification datasets, we have in Figure 4 in the upper row the experimental results without binary random column splits where we see that the choice of η is less important for increasing values of ε and overall not critical. The same holds for the lower row, where the results for DPSPNs learned with binary random column splits and fixed α are shown. For the 20 binary datasets, Figure 5 shows 10 datasets in the upper and 10 in the lower row, sorted by log-likelihood performance. Here, we see that overall a choice of $\eta \in \{0.4N, \ldots, 0.9N\}$ is preferable.

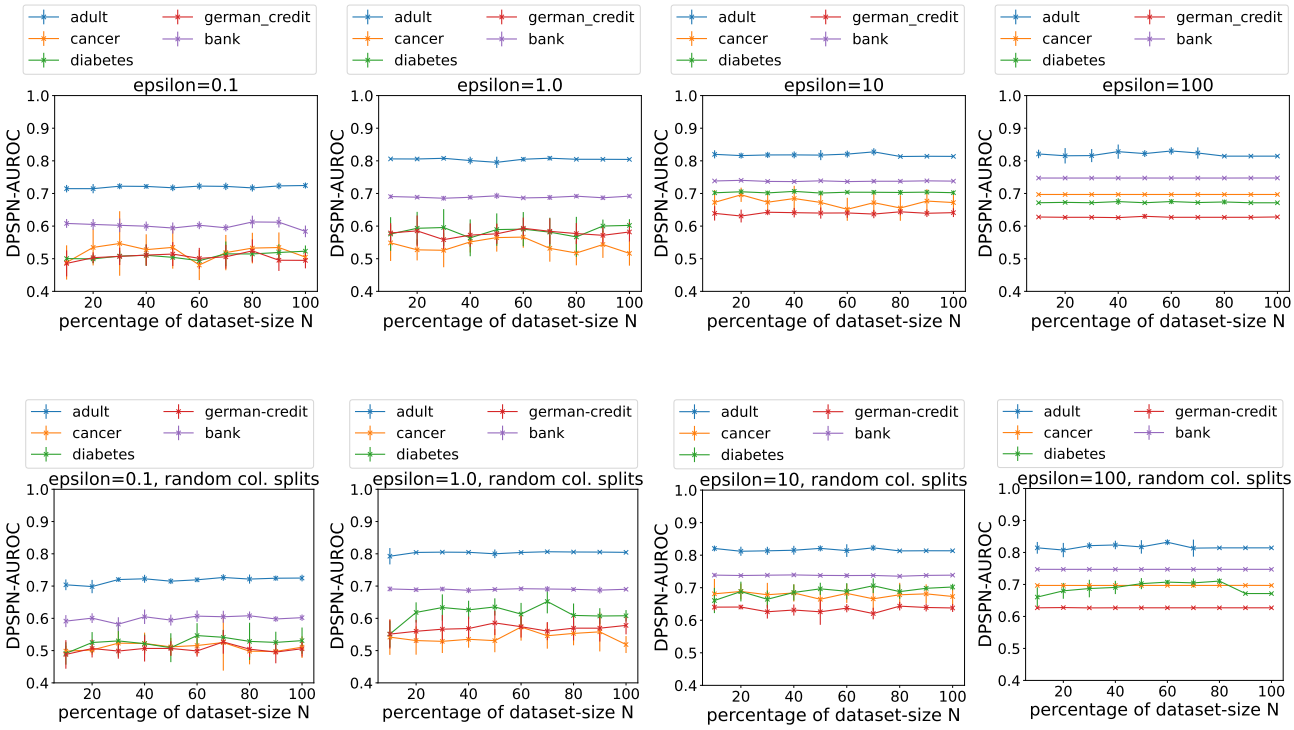

Figure 4. Influence of the parameter η (minimum of instances to split) for the classification datasets, no column splits and binary random column splits and for different epsilon values.

D. Utility Metrics.

Our holistic evaluation of synthetic generated data is based on Bowen & Snoke (2019) and Arnold & Neunhoeffer (2020) among others. Additional to evaluating the mean area under the receiver operating characteristic (AUROC) and the mean area under the precision-recall curve (AUPRC) over 5 different predictive models (Logistic Regression, Random Forest, Neural Network, Gaussian Naïve Bayes, Gradient Boosting Classifier), we also evaluated data utility in terms of the following metrics. As distributional distance metrics we applied the χ^2 test for categorical variables and Kolmogorov-Smirnov (KS) test for continuous variables. Each test was applied with conversion to p-values as scale-free distance measure and an average of the p-values for each variable gives the final score. For joint distribution metrics, we have the propensity score mean-squared error (pMSE) and the SPECKS (Synthetic data generation; Propensity score matching; Empirical Comparison via the Kolmogorov-Smirnov distance) score. The latter, applies the Kolmogorov-Smirnov (KS) distance to the predicted probabilities as the utility metric. Furthermore, we have two correlation metrics. The first, which is abbreviated with Reg-Coef, measures the standardized difference in coefficient values between a regression model fit on the original data and an regression model fit on the synthetic data. Here, we also calculate the confidence interval overlap (Reg-CI) for a single estimate on average of the two regressors.

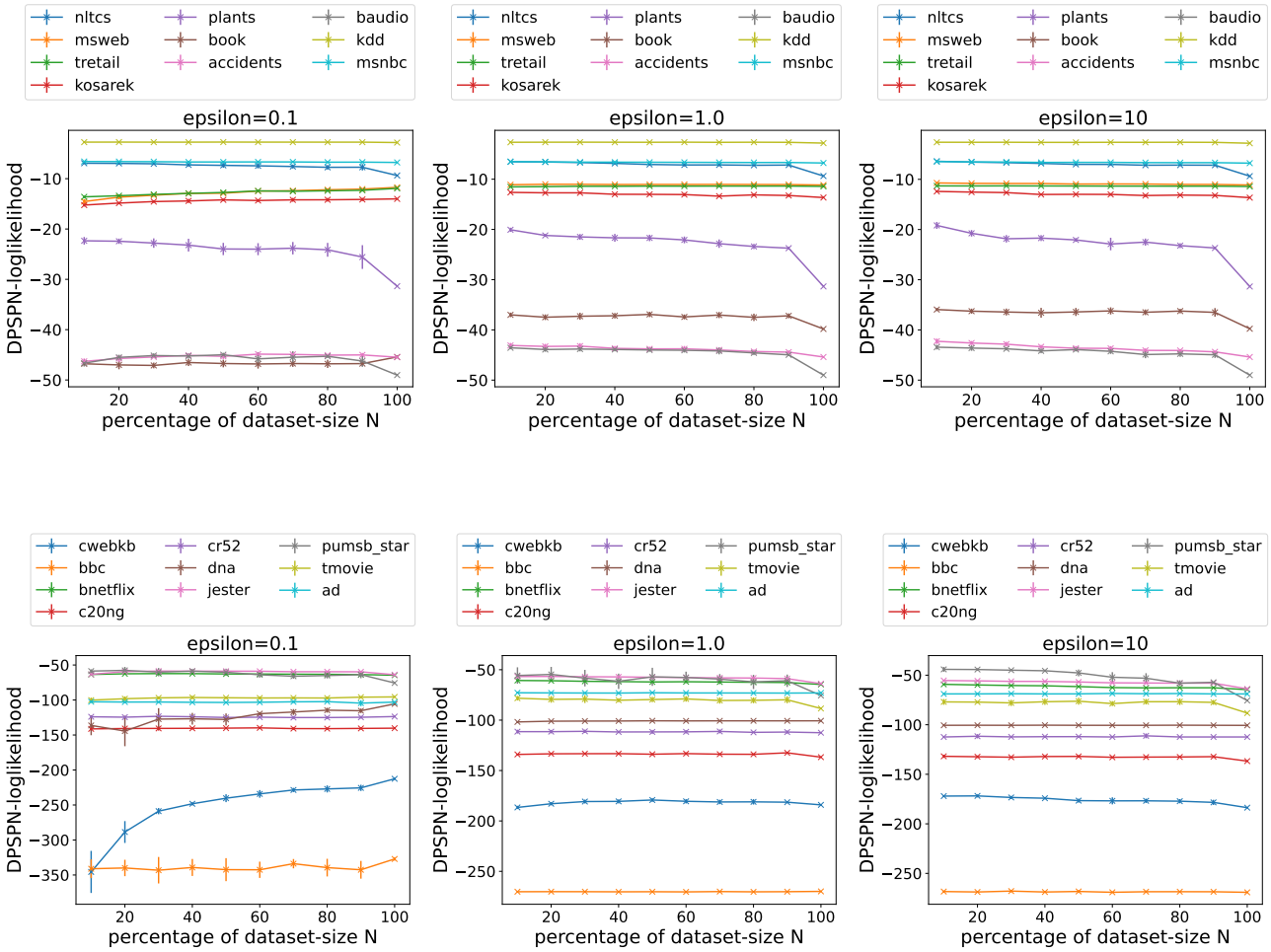

Figure 5. Influence of the parameter η (minimum of instances to split) for the binary datasets, binary random column splits and for different epsilon values.

E. Additional Results.

In Figure 6 we see how stable DPSPN, MST, PATE-GAN and PrivBayes (abbreviated as PrivBay) perform over 50 training runs on the diabetes dataset. In Table 5 we give the concrete values on which the graphs in Figure 2 are generated. In Table 6 to 10 we list the individual results for DP synthetic data generated by DPSPN, MST, PATE-GAN and PrivBayes (abbreviated as PrivBay)for 5 different predictive models – Logistic Regression, Random Forest, Neural Network, Gaussian Naïve Bayes and Gradient Boosting Classifier. We report the AUROC and AUPRC performance as well as mean performance. Figures 7 to 10 show additional visualizations of the utility of the generated DP data following the visualization protocol introduced by Bowen & Snoke (2019). In Table 11 we report further results for density estimation by DPSPNs over binary datasets.

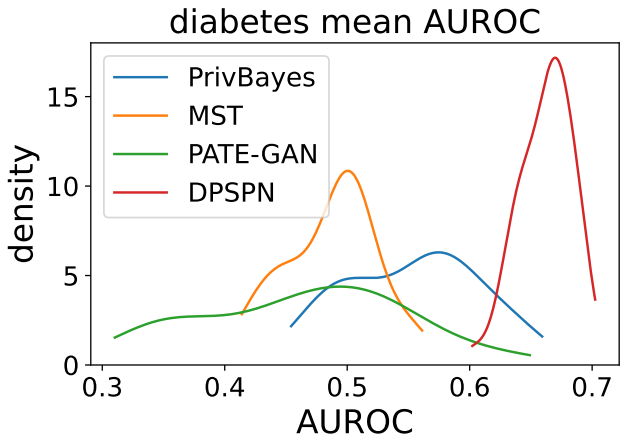

Figure 6. Distribution of test AUROC over 50 training runs for the diabetes dataset, mean over 5 models trained on DP synthetic data.

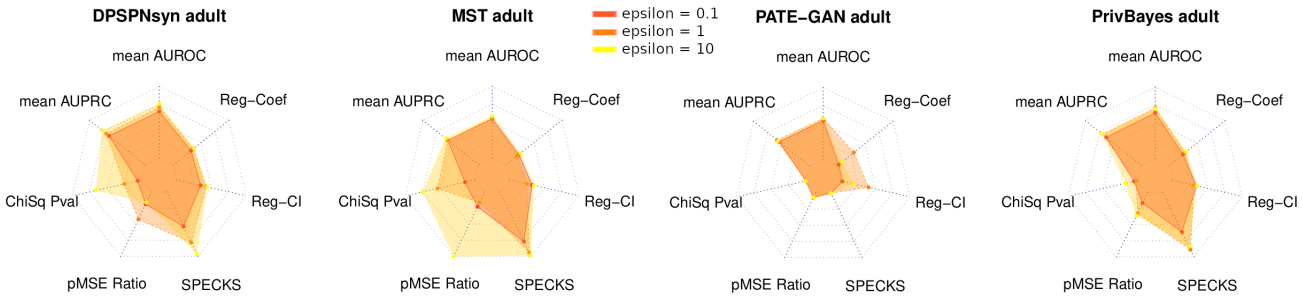

Figure 7. Utility summary charts for the adult dataset.

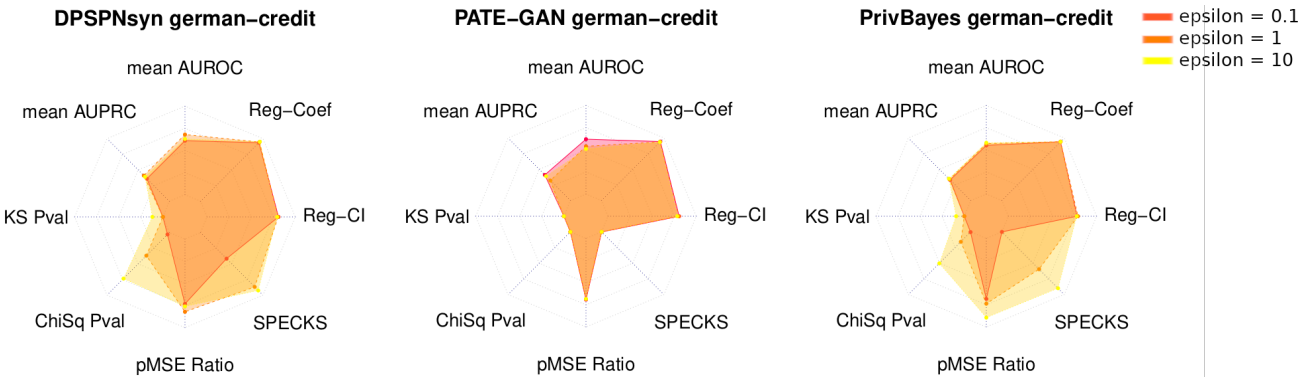

Figure 8. Utility summary charts for the german-credit dataset.

Table 5. Development of AUROC performance of DPSPNs as well as MSPNs trained on synthetically generated data without random columns splits and with binary random column splits, average of 10 runs.

		AUROC					
dataset	epsilon	MSPN	DPSPN	DPSPNrand	MSPNsyn	DPSPNsyn	DPSPNsynrand
adult	0.1	0.81	0.73	0.73	0.81	0.73	0.73
	0.5	0.81	0.8	0.8	0.81	0.8	0.8
	1	0.81	0.81	0.81	0.81	0.8	0.81
	5	0.81	0.82	0.82	0.81	0.81	0.81
	10	0.81	0.82	0.82	0.81	0.81	0.81
	50	0.81	0.83	0.83	0.81	0.81	0.81
	100	0.81	0.84	0.83	0.81	0.81	0.81
cancer	0.1	0.85	0.52	0.53	0.76	0.57	0.59
	0.5	0.85	0.55	0.52	0.76	0.65	0.62
	1	0.85	0.56	0.57	0.76	0.64	0.61
	5	0.85	0.67	0.66	0.76	0.68	0.69
	10	0.85	0.7	0.69	0.76	0.73	0.75
	50	0.85	0.7	0.7	0.76	0.8	0.8
	100	0.85	0.7	0.7	0.76	0.8	0.8
diabetes	0.1	0.67	0.56	0.55	0.63	0.55	0.54
	0.5	0.67	0.61	0.6	0.63	0.56	0.57
	1	0.67	0.65	0.65	0.63	0.58	0.58
	5	0.67	0.71	0.7	0.63	0.59	0.6
	10	0.67	0.7	0.71	0.63	0.59	0.58
	50	0.67	0.71	0.71	0.63	0.59	0.57
	100	0.67	0.71	0.71	0.63	0.59	0.58
german-credit	0.1	0.64	0.53	0.53	0.6	0.53	0.52
	0.5	0.64	0.56	0.56	0.6	0.56	0.55
	1	0.64	0.59	0.59	0.6	0.58	0.55
	5	0.64	0.65	0.64	0.6	0.58	0.58
	10	0.64	0.65	0.64	0.6	0.58	0.57
	50	0.64	0.63	0.63	0.6	0.58	0.58
	100	0.64	0.63	0.63	0.6	0.57	0.58
bank	0.1	0.75	0.61	0.61	0.73	0.62	0.62
	0.5	0.75	0.68	0.68	0.73	0.68	0.68
	1	0.75	0.69	0.69	0.73	0.69	0.7
	5	0.75	0.73	0.73	0.73	0.71	0.71
	10	0.75	0.74	0.74	0.73	0.72	0.72
	50	0.75	0.75	0.75	0.73	0.73	0.72
	100	0.75	0.75	0.75	0.73	0.73	0.73

Table 6. Performance comparison of 5 different predictive models trained on synthetic, tested on real in terms of AUROC and AUPRC. $(1, 10^{-6})$ -differentially private for the cancer dataset.

model	AUROC					AUPRC				
	Orig.	DPSPN	MST	PATE-GAN	PrivBay	Orig.	DPSPN	MST	PATE-GAN	PrivBay
Logistic Regression	0.88	1.0	0.98	0.82	0.8	0.61	0.97	0.59	0.24	0.24
Random Forests	0.98	0.98	0.98	0.44	0.62	0.68	0.71	0.71	0.06	0.14
Multi-layer Perceptron	0.48	0.46	0.38	0.5	0.47	0.06	0.05	0.05	0.08	0.05
Gaussian Naive Bayes	0.95	0.85	0.82	0.77	0.91	0.39	0.3	0.23	0.19	0.5
Gradient Boosting	0.98	0.89	0.95	0.27	0.81	0.68	0.27	0.45	0.09	0.16
Average	0.85	0.83	0.82	0.56	0.72	0.48	0.46	0.41	0.13	0.22

Table 7. Performance comparison of 5 different predictive models trained on synthetic, tested on real in terms of AUROC and AUPRC. $(1, 10^{-6})$ -differentially private for the diabetes dataset.

model	AUROC					AUPRC				
	Orig.	DPSPN	MST	PATE-GAN	PrivBay	Orig.	DPSPN	MST	PATE-GAN	PrivBay
Logistic Regression	0.74	0.74	0.61	0.69	0.68	0.38	0.42	0.29	0.35	0.37
Random Forests	0.75	0.77	0.67	0.69	0.64	0.38	0.43	0.35	0.38	0.27
Multi-layer Perceptron	0.74	0.25	0.25	0.29	0.31	0.37	0.14	0.14	0.15	0.15
Gaussian Naive Bayes	0.76	0.76	0.4	0.73	0.69	0.43	0.44	0.18	0.42	0.36
Gradient Boosting	0.79	0.77	0.6	0.73	0.64	0.42	0.46	0.26	0.4	0.28
Average	0.76	0.66	0.51	0.63	0.59	0.39	0.38	0.25	0.34	0.28

Table 8. Performance comparison of 5 different predictive models trained on synthetic, tested on real in terms of AUROC and AUPRC. $(1, 10^{-6})$ -differentially private for the german-credit dataset.

model	AUROC					AUPRC				
	Orig.	DPSPN	MST	PATE-GAN	PrivBay	Orig.	DPSPN	MST	PATE-GAN	PrivBay
Logistic Regression	0.72	0.75	0.56	0.69	0.67	0.49	0.5	0.32	0.44	0.47
Random Forests	0.67	0.67	0.59	0.63	0.59	0.4	0.4	0.4	0.37	0.35
Multi-layer Perceptron	0.48	0.52	0.52	0.58	0.52	0.27	0.29	0.31	0.31	0.31
Gaussian Naive Bayes	0.7	0.67	0.44	0.68	0.61	0.42	0.41	0.24	0.44	0.37
Gradient Boosting	0.69	0.64	0.62	0.61	0.58	0.4	0.38	0.37	0.38	0.35
Average	0.65	0.65	0.55	0.64	0.59	0.4	0.4	0.33	0.39	0.37

Table 9. Performance comparison of 5 different predictive models trained on synthetic, tested on real in terms of AUROC and AUPRC. $(1, 10^{-6})$ -differentially private for the adult dataset.

model	AUROC					AUPRC				
	Orig.	DPSPN	MST	PATE-GAN	PrivBay	Orig.	DPSPN	MST	PATE-GAN	PrivBay
Logistic Regression	0.88	0.86	0.83	0.79	0.88	0.73	0.66	0.62	0.58	0.69
Random Forests	0.9	0.88	0.59	0.66	0.88	0.77	0.7	0.43	0.39	0.71
Multi-layer Perceptron	0.89	0.89	0.58	0.73	0.88	0.75	0.73	0.43	0.54	0.72
Gaussian Naive Bayes	0.85	0.83	0.82	0.72	0.85	0.68	0.61	0.62	0.42	0.67
Gradient Boosting	0.92	0.89	0.61	0.74	0.89	0.8	0.74	0.44	0.51	0.74
Average	0.89	0.87	0.69	0.73	0.88	0.75	0.69	0.51	0.49	0.71

Table 10. Performance comparison of 5 different predictive models trained on synthetic, tested on real in terms of AUROC and AUPRC. $(1, 10^{-6})$ -differentially private for the bank dataset.

model	AUROC					AUPRC				
	Orig.	DPSPN	MST	PATE-GAN	PrivBay	Orig.	DPSPN	MST	PATE-GAN	PrivBay
Logistic Regression	0.83	0.66	0.67	0.71	0.67	0.83	0.67	0.6	0.69	0.67
Random Forests	0.92	0.75	0.48	0.51	0.69	0.94	0.71	0.49	0.53	0.68
Multi-layer Perceptron	0.83	0.63	0.52	0.39	0.73	0.82	0.61	0.54	0.43	0.72
Gaussian Naive Bayes	0.82	0.73	0.62	0.65	0.72	0.82	0.74	0.57	0.66	0.72
Gradient Boosting	0.92	0.77	0.49	0.6	0.72	0.93	0.76	0.53	0.59	0.7
Average	0.86	0.71	0.56	0.57	0.71	0.87	0.7	0.55	0.58	0.7

Table 11. SPN and DPSPN log-likelihood performance for the binary datasets and different epsilon values, average of 10 runs.

dataset	SPN	DPSPN		
		$\varepsilon = 0.1$	$\varepsilon = 1$	$\varepsilon = 10$
msnbc	-6.36	-6.59	-6.5	-6.46
msweb	-10.52	-11.73	-11.01	-10.72
plants	-16.11	-22.34	-20.07	-19.18
book	-34.88	-45.42	-36.92	-35.94
baudio	-40.34	-44.99	-43.56	-43.44
ad	-46.37	-102.33	-72.84	-68.44
jester	-53.96	-58.67	-56.66	-55.35
cr52	-93.33	-123.26	-111.1	-111.19
dna	-98.52	-105.55	-100.66	-100.44
bbc	-246.99	-327.1	-269.9	-267.88

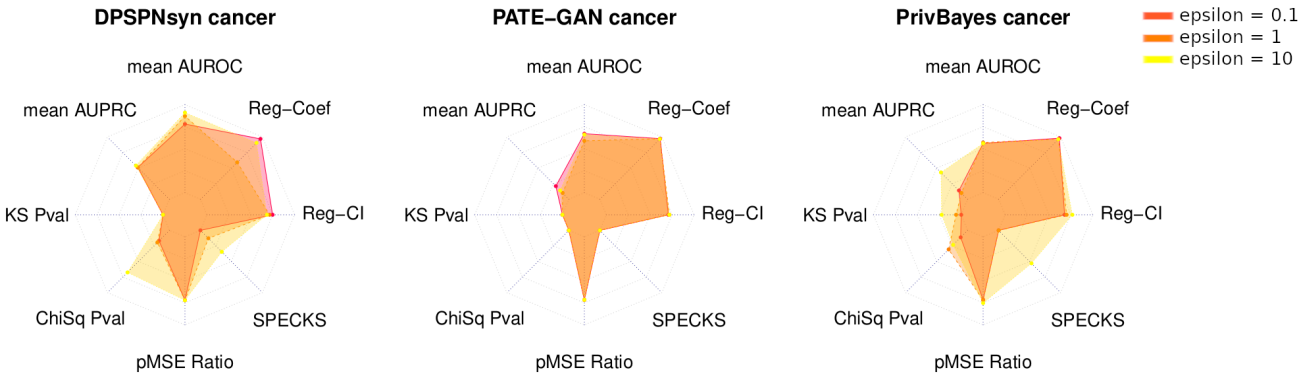

Figure 9. Utility summary charts for the cancer dataset.

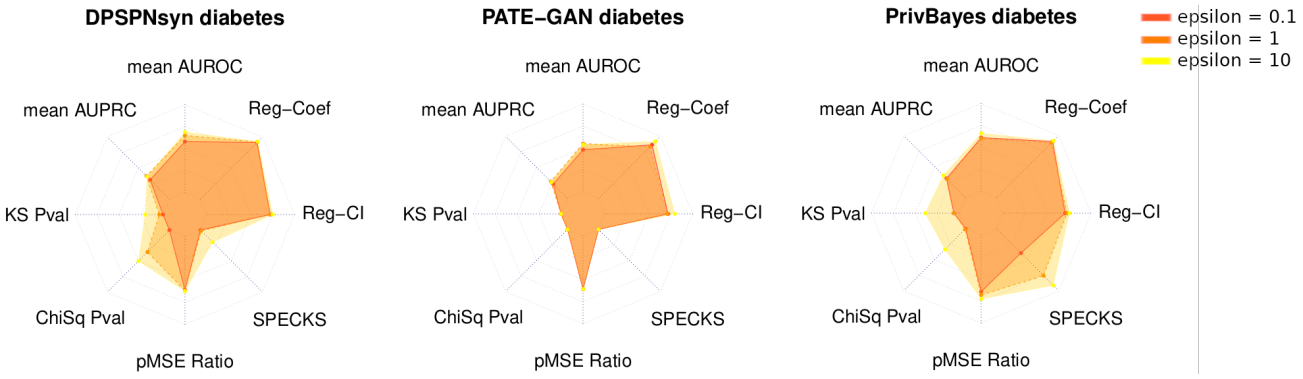

Figure 10. Utility summary charts for the diabetes dataset.