Complex & Intelligent Systems
https://doi.org/10.1007/s40747-024-01424-0

ORIGINAL ARTICLE

VertiBayes: learning Bayesian network parameters from vertically partitioned data with missing values

Florian van Daalen[1,2] · Lianne Ippel[2] · Andre Dekker[1] · Inigo Bermejo[1]

Received: 28 November 2023 / Accepted: 12 March 2024
© The Author(s) 2024

Abstract

Federated learning makes it possible to train a machine learning model on decentralized data. Bayesian networks are widely used probabilistic graphical models. While some research has been published on the federated learning of Bayesian networks, publications on Bayesian networks in a vertically partitioned data setting are limited, with important omissions, such as handling missing data. We propose a novel method called VertiBayes to train Bayesian networks (structure and parameters) on vertically partitioned data, which can handle missing values as well as an arbitrary number of parties. For structure learning we adapted the K2 algorithm with a privacy-preserving scalar product protocol. For parameter learning, we use a two-step approach: first, we learn an intermediate model using maximum likelihood, treating missing values as a special value, then we train a model on synthetic data generated by the intermediate model using the EM algorithm. The privacy guarantees of VertiBayes are equivalent to those provided by the privacy preserving scalar product protocol used. We experimentally show VertiBayes produces models comparable to those learnt using traditional algorithms. Finally, we propose two alternative approaches to estimate the performance of the model using vertically partitioned data and we show in experiments that these give accurate estimates.

Keywords Federated Learning · Bayesian network · Privacy preserving · Vertically partitioned data · Parameter learning · Structure learning

Introduction

Federated learning is a field that recently rose to prominence due to the increased focus on data-hungry techniques, privacy concerns and protection of the data [1, 2]. Using federated learning, it is possible to train a machine learning model without needing to collect the data centrally [1]. Since it rose to prominence, various techniques for training a model on centrally collected data have been adapted to be used on data that is either horizontally or vertically partitioned [2]. Data are said to be horizontally partitioned if multiple parties collect the same variables though from different individuals, e.g., two hospitals who want to build a model to predict heart failure. It is said to be vertically partitioned when multiple parties collect different variables about the same individuals, for example, data from a hospital and from a health insurance company where both parties have unique variables about the same patients.

A type of model that can benefit from federated learning is Bayesian networks. Bayesian networks are probabilistic graphical models that have been widely used in artificial intelligence [3–6]. They are popular because they can be built, verified, or improved, by combining data with existing expert knowledge. For example, medical doctors can manually create the network structure, ensuring it models already known dependencies correctly, while the conditional probability distributions are estimated from data. Thanks to its graphical representation and probabilistic reasoning, it is

✉ Florian van Daalen
f.vandaalen@maastrichtuniversity.nl

Lianne Ippel
gje.ippel@cbs.nl

Andre Dekker
andre.dekker@maastro.nl

Inigo Bermejo
i.bermejo@maastrichtuniversity.nl

1 Department of Radiation Oncology (MAASTRO), GROW School for Oncology and Reproduction, Maastricht University Medical Centre, Maastricht, The Netherlands

2 Methodology, Statistics Netherlands, Heerlen, The Netherlands

also a relatively intuitive model for non-technical personnel. This makes it very useful in scenarios where non-technical personnel needs to make decisions based on the model, for example when used as a tool to inform healthcare policies.

Existing literature on Bayesian networks in a federated setting

While research has been published on federated learning of Bayesian networks [7–10], publications on Bayesian networks trained on vertically partitioned data (also referred to as heterogeneous data in the literature) are limited. One proposed method [10] only deals with horizontally partitioned data, and the other approaches [7–9] are all only capable of handling two-party scenarios. In addition to this none of the proposed methods can handle missing values in the dataset.

Unfortunately, these two aspects are important in practical applications. Missing data are a common problem in real world scenarios this is especially true in federated scenarios where the different parties involved may have different data collection protocols and quality standards. In order to still have as large, and representative, a dataset as possible, records with missing data cannot be excluded.

The limitation to two-party scenarios is also a major downside in a federated setting. At its core federated learning attempts to combine data from as many data-sources as possible. Limiting algorithms to two-party scenarios runs directly counter to this goal.

Our contribution

In this article, we propose a novel method called VertiBayes to train Bayesian networks on vertically partitioned data, which can handle missing values as well as an arbitrary number of parties. In doing so we overcome the drawbacks the existing solutions have. This will allow us to train Bayesian networks in a vertically split federated setting, with an arbitrary number of parties, that are comparable to networks trained in a classical centrally trained setting.

The rest of the article is laid out as follows. First we will give some background information about Bayesian networks in general, and explain how these are trained in a classic scenario were all data is available centrally. Then we will describe our proposed method. After this we will describe the experimental setup we used to verify the federated model is similar to the centrally trained model. Followed by a discussion where we will go over aspects such as scalability and privacy concerns.

Bayesian networks

In this section, we will shortly explain how a Bayesian network is generally trained in a central setting.

Training a Bayesian network consists of two phases: structure learning and parameter learning. The first phase, structure learning, consists of determining the structure of the graph (i.e., the set of links between variables) and can be done either manually, using expert knowledge, or automatically, using algorithms such as K2 [11]. The second phase is the so-called parameter learning. In this phase, the conditional probability distributions (CPDs) for each node in the network are determined. Throughout this paper, we will focus on CPDs in the form of conditional probability tables (CPTs) as these are the most common form of CPD. In the next subsections, we will discuss how this is done in a centrally trained scenario. After which we will discuss how these methods need to be adapted for the federated scenario.

Structure learning

The structure of a Bayesian network can be either determined manually or learnt using an algorithm. Here, we focus on the latter, since the former does not involve data analysis. One of the most popular structure learning algorithms is K2, which performs a heuristic search for a viable structure by scoring potential parent nodes for a given node and step-wise adding the highest scoring parent [11]. The scoring function used in K2 is described in equation 1 below.

$$f(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \quad \prod_{k=1}^{r_i} a_{ijk}! \tag{1}$$

where pi_i is the set of parents of node x_i. q_i is the number of possible instantiations of the parents of x_i present in the data. r_i is the number of possible values the attribute x_i can take. a_{ijk} is the number of cases in the dataset where x_i has it's kth value and the parents are initiated with their jth combination. $N_{ij} = \sum_{k=1}^{r_i} a_{ijk}$, is the number of instances where the parents of x_i are initiated with their jth combination.

It is important to note that the resulting structure depends on the order in which nodes are introduced into the K2 algorithm. As such, it is possible to construct different structures for the same data.

Parameter learning

There are two relevant scenarios to consider when performing parameter learning: with and without missing data. When there is no missing data, CPDs can be learned using the maximum likelihood [12]. To calculate the maximum likelihood for an attribute X with a set of parents Y we simply have to calculate: $P(X = x_i|Y_i = y_i) = N_(x_i, y_i)/N_(y_i)$, where $N_(x_i, y_i)$ is the number of records where $X = x_i$ and $Y_i = y_i$ and $N_(y_i)$ is the number of records where $Y_i = y_i$. In the presence of missing data, the maximum likelihood for training a

Bayesian network is commonly estimated using algorithms such as Expectation Maximization (EM) [13, 14]. The EM algorithm consists of the following two steps repeated iteratively until convergence is reached:

1. Estimate the likelihood of your data using your current estimates of the probabilities.
2. Update your estimates.

To estimate the likelihood of the current estimates in the E-step the following equation needs to be solved:

$$E = \prod_{i=1}^{n} P(d_i) = \prod_{i=1}^{n} \prod_{j=1}^{p} P(x_{ij}|y_{ij}) \tag{2}$$

where n is the number of samples in the dataset, p is the number of nodes in the network, $P(d_i)$ is the likelihood of the $i-th$ sample, x_{ij} is the value of the $j-th$ node in the $i-th$ sample, and y_{ij} is the set of values of the parents of the $j-th$ node in the $i-th$ sample. The appropriate values $P(x_{ij}|y_{ij})$ need to be selected from the current estimate of the CPDs based on the attribute values of this particular sample.

It is important to note that since this algorithm is a hill climbing-type algorithm, it can get stuck in local optima. Therefore, it is good practice to run the algorithm several times with different random initializations and use the best result [12].

Method

VertiBayes

In this section we present our novel method VertiBayes and explain how it handles the various additional hurdles and concerns that arise in a vertically partitioned federated setting. First, we will discuss how to perform structure learning. In the second subsection, we will discuss parameter learning. After this we will discuss the time complexity of VertiBayes. Finally, we will discuss the impact a vertically split scenario has on classification and model validation for Bayesian networks, as well as provide several solutions to deal with the problems that arise.

Structure learning

As mentioned previously, structure learning can be done using the K2 algorithm. In this subsection, we will discuss how to adapt the K2 algorithm to a vertically partitioned scenario. To solve this equation, the following information needs to be collected:

1. The number of possible values for the attribute X.
2. The number of instances that fulfil $X = x_i$ and $Y = y_j$, where X is the child attribute, x_i is a given value for X, Y is the set of parent attributes, and y_j is a given set of assigned values to Y. This needs to be calculated for every possible set j.

The number of possible values of attribute X can be calculated trivially without revealing any important information to an external party in a vertically split federated setting as all relevant information is available locally at one party.

To calculate the number of instances that fulfil $X = x_i$ and $Y = y_i$ (the number of instances for each possible value of X for each possible configuration of X's parents) we have to calculate the number of instances that fulfil certain conditions across different datasets. There are different approaches we could utilize to solve this problem.

For example, we could leverage ϵ-differently privacy [15] to create a solution. This approach is relatively simple, however, it introduces noise, which can be problematic for smaller probabilities or for nodes with many parents, where a small amount of noise from each parent will eventually add up.

Alternatively we could attempt to solve it using homomorphic encryption [16]. Homomorphic encryption avoids adding any noise, but it is computationally expensive, especially as the K2 algorithm would require a fully homomorphic encryption scheme.

Finally a secret-sharing approach based in secure MultiParty Computation (MPC) [17] is an option. It is less computationally expensive than (full) homomorphic encryption, and does not introduce any noise. As such we propose to use this approach.

We propose to use the privacy preserving scalar product protocol to calculate the scalar product of vectors, one for each site, where each individual is represented as 1 or 0 depending on whether they fulfil the local conditions (in this case whether the child and parent nodes have the appropriate values). Earlier research has used this approach to calculate the information-gain when training a decision tree [18], which at its root, poses the same problem we face here. Additionally, this protocol also works in a hybrid setting, which allows our proposed method to be as versatile as possible.

Various variants of the privacy preserving scalar product protocol have been published [18–22]. Most of these focus on 2-party scenarios but variants do exist for N parties [23]. These methods have different advantages and disadvantages, such as different privacy guarantees and risks, different runtime complexities, and different communication cost overheads. Because of this, the preferred method will differ per scenario. A K2 implementation using one of these protocols will have the same privacy guarantees and risks but will pose no additional privacy concerns beyond those posed by the chosen protocol.

Parameter learning

During parameter learning, the actual CPDs will be calculated. There are two scenarios that need to be considered: with and without missing values. EM works under the assumption that data is missing at random or missing completely at random.

Without missing values As discussed earlier, parameter learning without missing values can be done by calculating the maximum likelihood for various attribute values. This means calculating for each node i, N_{ij}, the number of samples for each possible configuration of the parents of node i and N_{ijk} the number of samples for each possible configuration of the parents where the value is k, for each possible value of the node. N_{ijk} can be calculated by simply summing the various N_{ij} values. For the sake of performance it is advised to do this. These can be calculated using the scalar product protocol as explained earlier when describing the solution for K2. As such, performing parameter learning in a vertically split federated scenario with no missing values is not a problem and can be done without any significant additional privacy risks compared to the central variant beyond the risks involved in the scalar product protocol implementation used.

With missing values As mentioned in section 1.3.2 Expectation maximization requires that the appropriate values $P(x_{ij}|y_{ij})$ are selected from the current estimate of the CPDs based on the attribute values of this particular sample.

However, selecting the appropriate values $P(x_{ij}|y_{ij})$ can only be done when all child and parent node values are known. This is not possible in a privacy preserving setting if the child and parent nodes are spread over multiple parties. Conversely, even if it was possible to somehow select the appropriate values $P(x_{ij}|y_{ij})$, they may also never be revealed to anyone as it would be trivial to look up the parent and child node values in the CPD as the $P(x_{ij}|y_{ij})$ values will likely be unique.

It should be noted that a theoretical solution would be a layered approach combining homomorphic encryption with the privacy preserving scalar product protocol. However, due to the time complexity of each privacy preserving technique involved, the need to repeatedly execute the expectation step, and the fact this will need to be done for every single individual present in the training set, this is not practically viable.

Therefore, we conclude that the EM algorithm cannot be easily applied in a vertically split federated scenario without severe limitations. Instead, we propose the following three-step solution, which we have dubbed VertiBayes.

1. Treat“missing” as a valid value and train an intermediate Bayesian network using maximum likelihood on the training data.
2. Generate synthetic data (including “missing” values) using this intermediate Bayesian network.

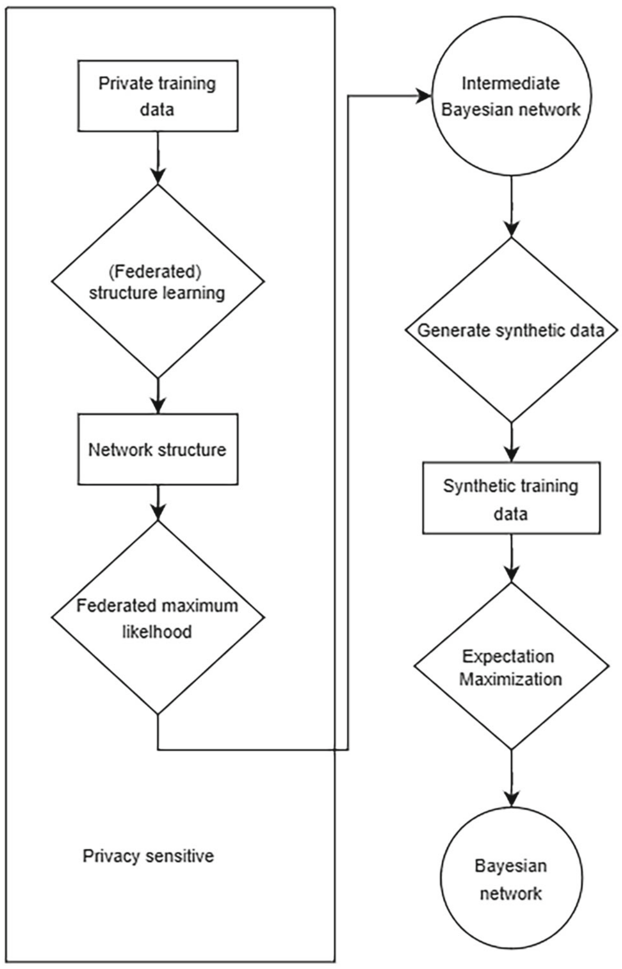

Fig. 1 Training process for VertiBayes

3. Train the final model on this synthetic data using the EM algorithm.

As discussed earlier, parameter learning in a vertically split federated setting without missing values is possible with the privacy guarantees provided by the privacy preserving scalar product protocol used. Generating synthetic data by using this intermediate model also does not add any additional privacy concerns compared to a centrally trained model, as this is a basic functionality of any Bayesian network. On the contrary, the final model has a reduced risk of data leak because it is trained on synthetic data [2, 24].

The proposed process, which is illustrated in Fig. 1, allows us to train a Bayesian network in a vertically split federated setting with missing values without any additional privacy concerns compared to a centrally trained model. However, it should be noted that it is possible that a loss of signal may occur due to the three-step approach. In the experiment section, we will test if our proposed method avoids this potential pitfall.

Time complexity when training a model in a federated setting

A major downside to federated learning is that the time complexity is usually considerably worse compared to the centralized setting. This is unavoidable due to the extra overhead created by communication as well as the increased complexity introduced by the privacy preserving mechanisms. In this subsection, we will discuss the time complexity of VertiBayes.

In our implementation, there are two important factors to take into account. The first is the number of parties n. This is a major bottleneck as the n-party scalar product protocol implementation we have used scales combinatorically in the number of parties.

The second important factor is the size of the CPDs that need to be calculated, as each unique probability that needs to be calculated requires a separate n-party scalar product protocol to be solved. As such, our implementation scales linearly in the number of probabilities that need to be calculated. The time complexity of various aspects for our implementation can be found in Table 1.

Important to note is that the population size is not a main driver of the runtime. A relatively simple network trained on a small dataset, but with a high number of unique attribute values will have a significantly longer runtime than a more complex network with few unique attribute values. This is because the overall time complexity is dominated by the number of scalar product protocols and subprotocols, which is independent of the population size, but dependent on the number of probabilities that need to be calculated.

Finally, it is important to note that there is ample room for parallelization to improve the running time as each scalar product protocol that is needed for VertiBayes is fully independent and can easily be run in parallel.

Federated classification and model validation

The process of using the model to classify new instances in a federated setting is itself a complex problem that depends strongly on the type of model used. In this subsection, we will discuss the methods that are available to classify an individual in a vertically partitioned setting using a Bayesian Network and the implication this has for the validation of the model.

Classification of new samples using a Bayesian Network in a vertically partitioned federated setting suffers from the same issues as the expectation step in the EM algorithm. To classify an instance from vertically partitioned data, we need to select the appropriate probabilities from the CPD. As discussed before, this is not viable while preserving privacy when parent and child nodes are split over multiple parties. This has major consequences for the validation of a new model in a federated setting.

As such, whenever possible the validation should be done using a publicly available dataset which avoids the need for privacy preserving measure during validation. If such a dataset is not available, we propose two different approaches, "Synthetic Cross-fold Validation" (SCV) and "Synthetic Validation Data Generation" (SVDG), to validate the model in a privacy preserving manner.

SCV uses the synthetic data generated by the intermediate Bayesian network as both training and validation data by executing the EM training using k-fold cross validation. However, it is possible that this results in overfitting on the synthetic data and therefore the performance estimate may be biased by the intermediate Bayesian network.

SVDG splits the private dataset into training and validation sets. It will then train a Bayesian network on the training set in a federated manner as normal. On the validation set, it will train a federated network using only the federated maximum likelihood approach. We can then use the Bayesian network trained on the validation set to generate a synthetic validation dataset. This approach reduces the risk of overfitting the previous approach suffered from but may lead to biased estimates if the synthetic validation set is not representative of the original validation set, for example because the test-fold was too small.

These approaches avoid leaking real data, but as mentioned, they may not be viable in practice. An illustration of the two new approaches can be seen in Fig. 2.

Table 1 Time complexity

Number of scalar product protocols	$O(m)$, where m is the number of unique parent-child value combinations for which a probability needs to be calculate
Number of scalar product subprotocols per protocol	$\frac{n!}{(x!(n-x)!))}$, for each x, $2 <= x <= n$, where n is the number of parties involved in the protocol
Number of multiplications per subprotocol	$O(p * n * (n-1))$, where p is the population size, and n is the number of parties involved in the protocol

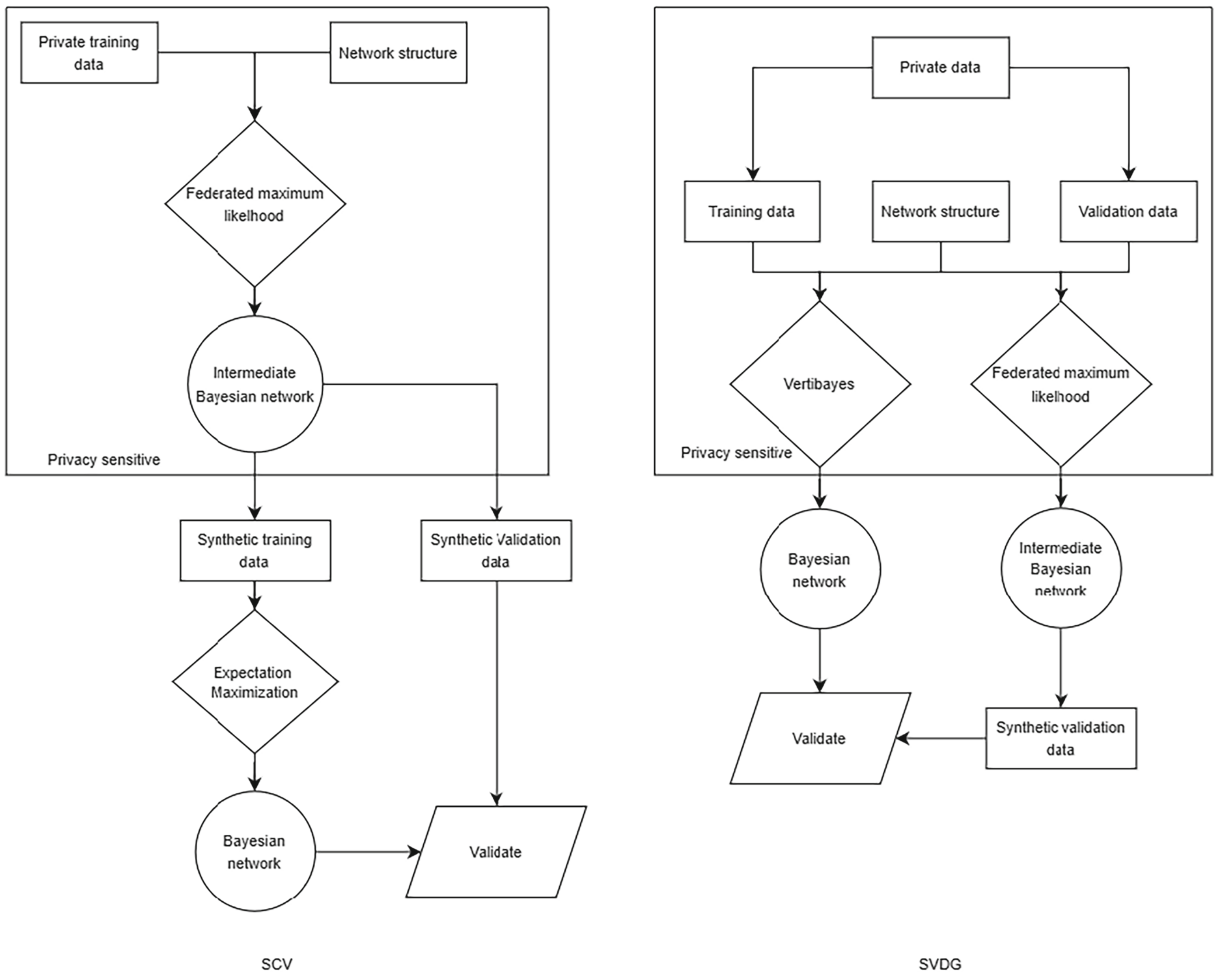

Fig. 2 Flow diagrams for proposed validation procedures SCV (left) and SVDG (right). SCV utilizes the first step of VertiBayes as normal, performing (federated) structure learning and federated maximum likelihood learning. The synthetic dataset generated during step 2 is split into a training and validation set. The training set is used as normal in step 3, which is then validated using the validation set. SVDG splits the data before performing any training into a training and validation set. The training set performs VertiBayes as normal. While we only run step 1 and 2 on the validation set. The synthetic data generated is then used to validate the model that was trained on the training set

Hyperparemeters

There are a number of choices which need to be made when initializing a new run of VertiBayes. These choices represent the various hyperparameters that can be set. The choices are as follows:

- Will structure learning be done using a predefined structure based on expert knowledge, or by utilizing the K2 algorithm.
- If K2 is used for the structure learning, what are the maximum amount of parents a node may have.
- Is discretization of continuous variables done using predefined bins based on expert knowledge, or utilizing an automatic approach. There are different strategies possible for automatic discretization which may have their own hyperparameters.
- What validation strategy is chosen from among the options in “Federated classification and model validation”.

The chosen structure learning approach can have a major impact on the performance of the resulting model. Similarly, the discretization approach can have a big impact as we will show in our experiments.

In the next section, we will perform experiments to validate that VertiBayes results in networks with similar performance as a centrally trained model. We will also verify if the two proposed approaches to validation give the same results as the validation on the public data, and if there are scenarios where they are inappropriate due to the aforementioned risks.

Experimental setup

In order to validate our proposed approach, we have implemented it in a combination of Java and Python[1] using Vantage6 [25] and ran a number of experiments. Vantage6 is an open-source infrastructure for privacy preserving federated learning which utilizes Docker. We compare the performance of our algorithm with a centrally trained Bayesian network using WEKA [26], a machine learning library written in Java.

The goal of the experiments is to show that the networks created by VertiBayes and the models created in a classic centralized scenario are the same. We did not compare results against other federated methods because (1) they cannot cope with missing data and (2) we use centralized learning as a baseline and want to show that VertiBayes performs equally well as a centralized approach

Structure learning

To validate our federated implementation of the K2 algorithm we ran experiments using the Iris [27], Asia [28], Alarm [29], and Diabetes [30] datasets. As K2 is deterministic and dependent on the order in which the nodes are put into the algorithm we ensured this was the same for both the federated learning and centralized learning model and then compared the resulting structures.

Parameter learning

In our experiments regarding parameter learning, we have used the Iris [27], Asia [28], Alarm [29], and Diabetes [30] datasets. In the case of the Iris dataset, we predict the “label” attribute, for Asia we predict “lung”, for Alarm we will be predicting “BP” and for Diabetes we will predict “outcome”. The Iris dataset contains 150 samples, the Diabetes dataset contains 768 samples, while the other two datasets contain data of 10.000 samples. The Asia and Alarm datasets come with a predefined structure. The Iris dataset uses a naive Bayes structure. The Diabetes dataset also uses a predefined structure.

[1] Our code can be found in the following two git repositories

- Main algorithm code: https://github.com/MaastrichtU-CDS/VertiBayes.
- Vantage6 wrapper code: https://github.com/MaastrichtU-CDS/VertiBayes_vantage6.

Both the Iris and Diabetes dataset contain continuous variables. For the sake of a fair comparison between the central and federated models, these were discretized into bins before starting our experiments, where each bin contains at least 10% of the total population as well as a minimum of 10 individuals. If the last bin cannot be made large enough to fulfil these criteria it is simply added to the previous bin. In the case of a dataset of less than 10 individuals, the bin will simply contain all possible values. For our experiments the bins were predefined alongside the predefined network structure, but the bins can also be generated on the fly during the training of the federated model using the same discretization strategy. The simplicity of this strategy allows it to be executed without needing additional privacy preserving mechanics. However, it should be noted that this is not the best possible discretizing strategy. For example, a model using the Minimum Description Length method (MDL) [31] for discretization, or utilizing expert knowledge, might outperform this setup.

Slight variations of this discretization strategy were used in preliminary experiments. However, we will not list the results of those variants here as they produced similar results.

To test the effect of missing values we have done experiments where we randomly set 0%, 5%, 10% and 30% of the values to missing. The performance of the models is measured by calculating the area under receiver operating characteristics curve (AUC). The centrally trained model is internally validated using 10-fold cross validation. The federated model is validated using the two different validation approaches described in the last section; it is also validated against a “public” central dataset, which is simply the left-out fold from the original dataset. All of these approaches use 10-fold cross validation. We also compare the Akaike information criterion (AIC) [32] values of the network for their original (private) training data. This is done to determine if there is any difference in the CPDs used by the two models. As mentioned before, the goal of the experiments is to get similar networks, as such we would expect the AUC and AIC of the networks produced by VertiBayes and the central approach to be similar. The AUC was chosen as a relevant metric because it is a powerful performance metric which naturally corrects for certain biases. For example, it does not have the same biases towards the majority class that accuracy has. AIC was chosen because it is a standard measure within Bayesian Network learning used to compare the complexity of the networks. This is important since we are not just interested in achieving similar performance, but wish to create similar networks.

For all of these experiments, the data are randomly partitioned over two parties by dividing the original dataset into two equally large sets of attributes.

Scaling in the number of dataparties

To illustrate VertiBayes works when dealing with multiple parties the aforementioned parameter learning experiment was repeated for 3–8 parties using the Asia dataset. As this dataset contains 8 attributes this means the number of attributes varied from 4 to 1 attribute per party.

Results

The results of our experiments can be found in this section. First we will discuss the results of the structure learning experiments. After that we will cover the experiments regarding parameter learning.

Structure learning

The federated implementation of the K2 algorithm consistently resulted in the same structure as the centrally trained model for all datasets.

Parameter learning

Table 2 shows the results of our parameter learning experiments. It lists the AUC for the centrally trained model, as well as the AUC's for the various (federated) validation schemes. The AUC's of the federated model are listed in bold if the difference with the central model is larger than 0.05. The AIC is shown for the centrally trained model, this is compared to the AIC for the federated model trained. The difference between the central and federated AIC is listed in bold if the difference is at least 5%. An AIC closer to 0 is better. A negative percentage in the AIC difference column indicates the federated model performed better. The results are shown per dataset for differing levels of missing data; no missing data, 5% missing data, 10% missing data, and 30% missing data. VertiBayes showed similar performance to the centrally trained model in all scenarios.

Time complexity

To illustrate the time complexity we kept track of the runtime of the parameter learning experiments. The results can be found in Fig. 3. The relative runtimes are plotted versus population size, number of attributes, and total size of the CPDs that need to be calculated during the Maximum Likelihood stage of VertiBayes. As can be clearly seen in these graphs the performance depends mostly on the size of the CPDs, and is virtually independent from population size. Number of attributes does correlates with a longer runtime because more attributes often implies more probabilities will need to be calculated to fill all CPDs However, since the size of the CPDs also depend on how connected the network is and how many values each attribute can take it does not correlate perfectly.

Scaling in the number of dataparties

The results of our experiments with the Asia dataset using multiple parties can be found in Table 3. The number of parties has no meaningful impact on the performance of VertiBayes. The minor differences are due to the inherent variation introduced by several random factors within VertiBayes, such as the random nature of the synthetic data that is generated during the second step of VertiBayes.

Discussion

In this paper, we have proposed a novel method to train Bayesian networks in a federated setting using vertically partitioned data with missing values. The results of our experiments have shown that it is possible to perform both structure and parameter learning of Bayesian networks in such a setting with reasonable accuracy. Structure learning can be performed by adapting any of the existing structure learning algorithms to use a secure multiparty computation algorithm. In this study, we used a protocol within the K2 algorithm, but the same approach could be applied to other score-based algorithms or even constraint-based algorithms such as the PC algorithm [33]. Parameter learning requires one of two approaches. When there is no missing data present, the scalar product protocol can be used directly to compute the maximum likelihood, or when missing data is present the three-step solution described in "VertiBayes" using the EM algorithm can be applied. We will now discuss the performance of VertiBayes compared to a centrally trained model as well as the limitations in terms of scalability. Lastly, we will discuss the sensitive information that may be leaked by any Bayesian network and the limitations this brings in a federated setting.

Model performance and validation

Our experiments show that the resulting models produced by VertiBayes are comparable to the centrally trained models. As such, there is generally no meaningful difference in terms of AUC or AIC. The added privacy guarantees make it possible to train a model in a vertically partitioned setting. This takes away certain barriers with respect to data-sharing, allowing models to be trained on larger sets of data, which contains data that would have been inaccessible in a centralized setting. Utilizing this normally inaccessible data should lead to improved models in real life scenarios.

Table 2 Results of the experiments.

Dataset		AUC Training method				AIC Training method		
		Centralized learning	Federated learning			Centralized learning	Federated learning	AIC difference
	Missing data %	Public validation	Public validation	SCV validation	SVDG validation			
Alarm population size: 10,000	0	0.91	0.91	0.91	0.91	−340571	−318469	**−6.49%**
	5	0.88	0.89	0.88	0.89	−315856	−313612	−0.71%
	10	0.85	0.86	0.86	0.86	−340823	−350576	2.86%
	30	0.76	0.76	0.76	0.76	−444297	−444866	0.13%
Asia population size: 10,000	0	1.00	1.00	1.00	1.00	−22555	−22559	0.02%
	5	0.76	0.76	0.76	0.76	−23430	−23395	−0.15%
	10	0.69	0.7	0.7	0.7	−24105	−24090	−0.06%
	30	0.58	0.59	0.58	0.59	−25517	−25613	0.37%
Diabetes population size: 768	0	0.8	0.77	**0.95**	0.79	−13593	−14407	**5.99%**
	5	0.79	**0.74**	**0.92**	0.76	−13699	−14556	**6.25%**
	10	0.75	0.72	**0.90**	0.73	−13761	−14408	4.70%
	30	0.61	0.57	**0.80**	0.6	−14736	−15136	2.71%
Iris population size: 150	0	0.99	0.98	1.00	1.00	−1036	−1022	−1.33%
	5	0.97	0.96	0.99	0.97	−1243	−1176	**−5.37%**
	10	0.9	0.89	0.95	0.92	−1491	−1022	**−31.49%**
	30	0.98	0.99	1.00	0.99	−1099	−1381	**25.67%**

The experiments are grouped per dataset. The AUC is represented for the centrally trained model, as well as the two federated validation schemes; “Synthetic Cross-fold Validation” (SCV) and “Synthetic Validation Data Generation” (SVDG). AUC values are listed in bold if they differ more than 0.05 from the central model. An AIC closer to 0 is better, a negative percentage in the AIC difference column indicates the federated model was better. This value is listed in bold if the difference between the federated AIC and the central AIC is at least 5%

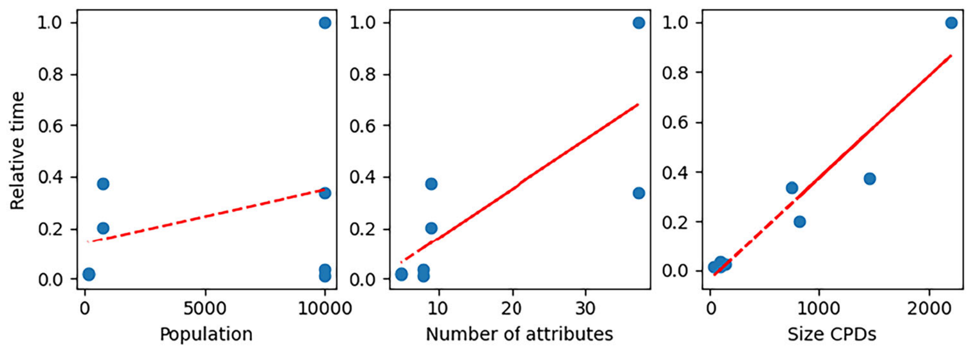

Fig. 3 Relative runtime of VertiBayes using various datasets plotted versus population size, number of attributes, and total number of probabilities that need to be calculated during the Maximum Likelihood stage

Table 3 Results of the repeated experiments with multiple parties for the Asia dataset

		AUC				
	Number of parties	Public validation	SCV validation	SVDG validation	AIC score	Running time MS
Asia Missing data: 0%	2	0.97	1.00	1.00	−22575	142292
	3	0.98	1.00	1.00	−22564	144124
	4	0.98	1.00	1.00	−22642	145343
	5	0.98	1.00	1.00	−22600	144756
	6	0.97	1.00	1.00	−22570	142854
	7	0.99	1.00	1.00	−22530	144551
	8	0.98	1.00	1.00	−22488	145143
Asia Missing data: 10%	2	0.70	0.71	0.70	−23888	426849
	3	0.70	0.70	0.70	−23837	426379
	4	0.70	0.69	0.71	−23837	425364
	5	0.70	0.70	0.71	−23918	427819
	6	0.70	0.71	0.71	−24042	427391
	7	0.69	0.69	0.71	−23871	432755
	8	0.70	0.70	0.70	−24073	412467

Furthermore, the experiments show that it is possible to validate the model in a privacy-preserving manner despite it being impossible to efficiently classify an individual in a privacy-preserving manner.

It is however important to note that in certain edge-cases some validation approaches can show unreliable results. For example, the SVDG approach can cause problems when the test-folds are too small and the bins are generated on the fly while training, as opposed to working with pre-defined bins. If there are not enough individuals in the test-set to create multiple accurate bins on the fly this strategy will result in a loss of information. This was most notable when running the preliminary experiments with the Iris dataset, which is quite small.

Similarly, the model ran into problems using the SCV approach whenever the CPDs become too large because a node has multiple parents with a significant amount of bins. This is notable in the results of the Diabetes model. Certain nodes with multiple parents would end up with CPDs that contained more cells than there are individuals in the dataset. This led to an overestimation of the AUC if the SCV approach was used, as it overfits on the training data. However, this was not an issue when utilizing the SVDG approach, due to the stronger separation between training and validation data. Using larger bins can alleviate this problem to some extent. However, the bins cannot be made arbitrarily large as this will eventually cause a loss of information. Using expert-knowledge based discretization strategies tailored to each dataset, or a better automatic discretization strategy such as the MDL method mentioned earlier, would help avoid these problems.

These problems of overfitting and loss of information, show that it is extremely important to have an appropriate discretization strategy. So long as the potential pitfalls surrounding discretization are addressed, VertiBayes can be used to train and validate a Bayesian network in a vertical federated setting.

Scalability

Any algorithm that is adapted to a federated setting will be slower than the central counterpart due to the overhead caused by the protocols used to protect privacy. Our experiments confirmed the potential issues we brought up in “Time complexity when training a model in a federated setting” when we discussed the theoretical time complexity.

The effects of population-size proved to be negligible. The effects of the complexity of the network structure, that is to say the number of nodes and links within the network, is only relevant in so much that it creates more probabilities to calculate. As expected, the size of the CPDs that had to be calculated had the greatest effect on the total runtime.

The number of parties also proved to not be very impactful. This intuitively makes sense as the bottleneck for VertiBayes lies in the calculation of the CPDs. When calculating the CPD for a particular attribute we can easily deduce the maximum parties involved in this calculation. For example $P(X|Y)$ will involve at most 2 parties in a vertical partitioned setting as it only involves 2 attributes. Similarly, calculating $P(X|Y, Z)$ will involve at most 3 parties. This means that the effect of the number of parties is naturally limited depending on the maximum amount of parents a node has in the network structure.

This does mean that there are practical limitations to using VertiBayes, as it may take too long to train a large or complex network. However, it should be noted that in certain settings a longer runtime might still be acceptable. For example, it is perfectly acceptable that training a model for use in a clinical setting takes an extended amount of time.

Sensitive information in published Bayesian networks

Publishing a Bayesian network, or any machine learning model, will reveal certain information about the training data, regardless of how the network is trained. When publishing a Bayesian network two important aspects will be revealed: the network structure and the CPDs. The network structure will only reveal which conditional dependencies exist amongst attributes, which is not sensitive data in most scenarios. The CPDs on the other hand, can potentially be used to reconstruct individual level data from the training-set, when the probabilities in the CPD’s are based on one, or a few individuals. An effective countermeasure is using k-anonymity [34] to ensure that each probability in the CPDs represents a minimum amount of samples and that no probabilities of 0 or 1 are present in the CPDs. Such probabilities make it considerably easier to deduce individual level data, and they can also lead to artefacts when using the network.

Lastly, a public Bayesian network can be used by one of the parties that participated in the training to guess (although not reconstruct) the data of the other parties based on their own data. Similarly, any third party with partial data can use the final model to estimate the missing values in his dataset. This is unavoidable and it should be taken into consideration when decisions are made about which models are to be made public.

These concerns imply that there are practical limits to what privacy preserving techniques should aim for. Trying to prevent any and all privacy issues using privacy preserving techniques during the training phase is futile when models are made public as the models themselves will always reveal some information.

Adapting the structure learning approach

In this article we choose to utilize the K2 algorithm to learn the network structure. Other approaches exist as well, these can be score-based [35] or constraint based [36]. Extending VertiBayes to utilize one of these alternatives is possible. At its core VertiBayes uses the privacy preserving n-scalar product protocol to calculate simple statistics such as the maximum likelihood. Any score based on constraint based structure learning algorithm that can is based on similar simple statistics can be calculated in a privacy preserving manner in a similar way.

Conclusion

In this paper, we have proposed a novel approach to train Bayesian network parameters from a vertically partitioned data with and without missing values. This method can deal with an arbitrary number of parties, only limited by the runtime. We have shown that there are no additional privacy risks compared to a centrally trained model beyond the ones presented by the specific privacy preserving scalar product protocol implementation used. Our experiments show there are no meaningful differences in performance between models trained with VertiBayes and models trained centrally, provided continuous variables are adequately discretized. They also show it is possible to estimate the performance of a model with vertically partitioned data with a reasonable accuracy. As such, VertiBayes is a useful tool for training Bayesian networks in a vertically partitioned setting. Utilizing Bayesian networks in a vertically partitioned setting, will unlock normally inaccessible data, which will lead to improved models in real life scenarios.

Future work

When using the model in a federated setting with a vertical split, it is currently not possible to efficiently classify or predict a new instance in a privacy preserving manner using a Bayesian network. It would be beneficial if a solution for this was found and implemented. In addition to this, VertiBayes could be improved by implementing more advanced discretization methods, such as MDL, in a vertically partitioned setting as our current implementation relies either on a very basic automatic discretization approach or the use of experts, which may not be the best discretization approach possible. Additionally, the impact of different missing data mechanisms on our proposed approach should be investigated. In this article we used data that missed completely at random, however, we did not look at other missing mechanisms, such as missing at random. It would be worthwhile to investigate whether this significantly influences the performance of the resulting model. Lastly, it would be beneficial to run experiments in different real life scenarios to verify how VertiBayes scales in practice, especially with regards to the number of parties participating. As mentioned in Sect. 4.2 we do not expect major in realistic scenarios, but this should be verified.

Author Contributions Not applicable.

Funding This research received funding from the Netherlands Organization for Scientific Research (NWO): Coronary ARtery disease: Risk estimations and Interventions for prevention and EaRly detection (CARRIER): project nr. 628.011.212.

Data Availability Not applicable.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent to publication Not applicable.

Code availability The code is available and can be found here: Main algorithm code: https://github.com/MaastrichtU-CDS/VertiBayes, Vantage6 wrapper code: https://github.com/MaastrichtU-CDS/VertiBayesvantage6.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Li L, Fan Y, Tse M, Lin K-Y (2020) A review of applications in federated learning. Comput Indus Eng 149:106854. https://doi.org/10.1016/j.cie.2020.106854
2. Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, D'Oliveira RGL, Eichner H, Rouayheb SE, Evans D, Gardner J, Garrett Z, Gascón A, Ghazi B, Gibbons PB, Gruteser M, Harchaoui Z, He C, He L, Huo Z, Hutchinson B, Hsu J, Jaggi M, Javidi T, Joshi G, Khodak M, Konecný J, Korolova A, Koushanfar F, Koyejo S, Lepoint T, Liu Y, Mittal P, Mohri M, Nock R, Özgür A, Pagh R, Qi H, Ramage D, Raskar R, Raykova M, Song D, Song W, Stich SU, Sun Z, Suresh AT, Tramèr F, Vepakomma P, Wang J, Xiong L, Xu Z, Yang Q, Yu FX, Yu H, Zhao S (2021) Advances and open problems in federated learning. Found Trends(R) Mach Learn (Now Publishers, Inc.) 14(1–2):1–210. https://doi.org/10.1561/2200000083
3. Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco
4. Wang H, Núñez A, Liu Z, Zhang D, Dollevoet R (2019) A Bayesian network approach for condition monitoring of high-speed railway catenaries. IEEE Trans Intell Transport Syst 21(10):4037–4051
5. Chen R, Lu Y, Witherell P, Simpson TW, Kumara S, Yang H (2021) Ontology-driven learning of Bayesian network for causal inference and quality assurance in additive manufacturing. IEEE Robot Autom Lett 6(3):6032–6038
6. McLachlan S, Dube K, Hitman GA, Fenton NE, Kyrimi E (2020) Bayesian networks in healthcare: distribution by medical condition. Artif Intell Med 107:101912
7. Yang Z, Wright RN (2006) Privacy-preserving computation of Bayesian networks on vertically partitioned data. IEEE Trans Knowl Data Eng 18(9):1253–1264. https://doi.org/10.1109/TKDE.2006.147
8. Wright R, Yang Z (2004) Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In: Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '04, pp 713–718, New York. https://doi.org/10.1145/1014052.1014145 (2004)
9. Yang Z, Wright RN (2005) Improved privacy-preserving Bayesian network parameter learning on vertically partitioned data. In: 21st international conference on data engineering workshops (ICDEW'05), Tokyo, pp 1196–1196. https://doi.org/10.1109/ICDE.2005.230
10. Ng I, Zhang K (2022) Towards federated Bayesian network structure learning with continuous optimization. In: Proceedings of The 25th international conference on artificial intelligence and statistics, pp 8095–8111. ISSN: 2640-3498. https://proceedings.mlr.press/v151/ng22a.html. Accessed 26 Jan 2023
11. Cooper GF, Herskovits E (1992) A Bayesian method for the induction of probabilistic networks from data. Mach Learn 9(4):309–347. https://doi.org/10.1007/BF00994110
12. Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques—adaptive computation and machine learning
13. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B (Methodol) 39(1):1–38 (Royal Statistical Society, Wiley)
14. Lauritzen SL (1995) The EM algorithm for graphical association models with missing data. Comput Stat Data Anal 19(2):191–201. https://doi.org/10.1016/0167-9473(93)E0056-A
15. Dwork C, Roth A (2014) The algorithmic foundations of differential privacy. Found Trends(R) Theor Comput Sci 9(3):211–407. https://doi.org/10.1561/0400000042

16. Parmar PV, Padhar SB, Patel SN, Bhatt NI, Jhaveri RH (2014) Survey of various homomorphic encryption algorithms and schemes. Int J Comput Appl 91(8):26–32. https://doi.org/10.5120/15902-5081
17. Yao AC (1982) Protocols for secure computations. In: 23rd annual symposium on foundations of computer science (SFCS 1982), pp 160–164 (1982). https://doi.org/10.1109/SFCS.1982.38 . ISSN: 0272-5428
18. Du W, Zhan Z (2002) Building decision tree classifier on private data. In: Proceedings of the IEEE international conference on privacy, security and data mining, CRPIT '14, vol 14, pp 1–8, AUS (2002)
19. Du W, Atallah MJ (2001) Privacy-preserving cooperative statistical analysis. In: Seventeenth annual computer security applications conference, New Orleans, pp 102–110. https://doi.org/10.1109/ACSAC.2001.991526
20. Atallah MJ, Du W (2001) Secure multi-party computational geometry. In: Goos G, Hartmanis J, Leeuwen J, Dehne F, Sack J-R, Tamassia R (eds) Algorithms and data structures, vol 2125, pp 165–179. Springer, Berlin. https://doi.org/10.1007/3-540-44634-6_16 (series title: lecture notes in computer science)
21. Goethals B, Laur S, Lipmaa H, Mielikäinen T (2005) On private scalar product computation for privacy-preserving data mining. In: Hutchison D, Kanade T, Kittler J, Kleinberg JM, Mattern F, Mitchell JC, Naor M, Nierstrasz O, Pandu Rangan C, Steffen B, Sudan M, Terzopoulos D, Tygar D, Vardi MY, Weikum G, Park C-S, Chee S (eds) Information security and cryptology—ICISC 2004, vol 3506, pp 104–120. Springer, Berlin. https://doi.org/10.1007/11496618_9 (series title: lecture notes in computer science)
22. Vaidya J, Clifton C (2002) Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '02, pp 639–644, New York (2002). https://doi.org/10.1145/775047.775142
23. Daalen F, Ippel L, Dekker A, Bermejo I (2023) Privacy preserving n-party scalar product protocol. IEEE Trans Parallel Distrib Syst 34(4):1060–1066 (2023) https://doi.org/10.1109/TPDS.2023.3238768 (conference name: IEEE Transactions on Parallel and Distributed Systems)
24. Abay NC, Zhou Y, Kantarcioglu M, Thuraisingham B, Sweeney L (2019) Privacy preserving synthetic data release using deep learning. In: Berlingerio M, Bonchi F, Gärtner T, Hurley N, Ifrim G (eds) Machine learning and knowledge discovery in databases, vol 11051, pp 510–526. https://doi.org/10.1007/978-3-030-10925-7_31 (series title: lecture notes in computer Science)
25. Moncada-Torres A, Martin F, Sieswerda M, Van Soest J, Geleijnse G (2020) VANTAGE6: an open source priVAcy preserviNg federaTed leArninG infrastructurE for secure insight eXchange. AMIA. Annual symposium proceedings. AMIA symposium 2020, pp 870–877 (2020)
26. Frank E, Witten IH, Hall MA (2016) Data mining, 4th edn. Practical machine learning tools and techniques|guide books
27. De Marsico M, Nappi M, Riccio D, Wechsler H (2015) Mobile iris challenge evaluation (MICHE)-I, biometric iris dataset and protocols. Pattern Recognit Lett 57:17–23. https://doi.org/10.1016/j.patrec.2015.02.009
28. Lauritzen SL, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems. J R Stat Soc Ser B (Methodol) 50(2):157–194. https://doi.org/10.1111/j.2517-6161.1988.tb01721.x
29. Beinlich IA, Suermondt HJ, Chavez RM, Cooper GF (1989) The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In: AIME 89, pp 247–256. : Springer, Berlin. https://doi.org/10.1007/978-3-642-93437-7_28
30. Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS (1988) Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the annual symposium on computer application in medical care, pp 261–265
31. Fayyad UM, Irani KB (1993) Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI, pp 1022–1029
32. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19(6):716–723. https://doi.org/10.1109/TAC.1974.1100705 (conference name: IEEE transactions on automatic control)
33. Spirtes P, Glymour CN, Spirtes P, Glymour C (1991) An algorithm for fast recovery of sparse causal graphs. Soc Sci Comput Rev 9:62–72
34. Sweeney L (2002) k-Anonymity: a model for protecting privacy. Int J Uncertain Fuzzin Knowl-Based Syst 10(05):557–570. https://doi.org/10.1142/S0218488502001648 (publisher: World Scientific Publishing Co)
35. Ramirez-Hereza P, Ramos D, Toledano DT, Gonzalez-Rodriguez J, Ariza-Velazquez A, Doncel N (2023) Score-based Bayesian network structure learning algorithms for modeling radioisotope levels in nuclear power plant reactors. Chemomet Intell Lab Syst 237:104811
36. Gonzales C, Journe A, Mabrouk A (2021) Constraint-based Bayesian network structure learning using uncertain experts' knowledge. In: Thirty-fourth international Florida Artificial Intelligence Research Society conference, vol 34 (2021)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.