Probabilistic Dataset Reconstruction from Interpretable Models

1st Julien Ferry
LAAS-CNRS, Université de Toulouse, CNRS
Toulouse, France
jferry@laas.fr

2nd Ulrich Aïvodji
École de Technologie Supérieure
Montréal, Canada
Ulrich.Aivodji@etsmtl.ca

3rd Sébastien Gambs
Université du Québec à Montréal
Montréal, Canada
gambs.sebastien@uqam.ca

4th Marie-José Huguet
LAAS-CNRS, Université de Toulouse, CNRS, INSA
Toulouse, France
huguet@laas.fr

5th Mohamed Siala
LAAS-CNRS, Université de Toulouse, CNRS, INSA
Toulouse, France
msiala@laas.fr

***Abstract*—Interpretability is often pointed out as a key requirement for trustworthy machine learning. However, learning and releasing models that are *inherently interpretable* leaks information regarding the underlying training data. As such disclosure may directly conflict with privacy, a precise quantification of the privacy impact of such breach is a fundamental problem. For instance, previous work [1] have shown that the structure of a decision tree can be leveraged to build a probabilistic reconstruction of its training dataset, with the uncertainty of the reconstruction being a relevant metric for the information leak. In this paper, we propose of a novel framework generalizing these probabilistic reconstructions in the sense that it can handle other forms of interpretable models and more generic types of knowledge. In addition, we demonstrate that under realistic assumptions regarding the interpretable models' structure, the uncertainty of the reconstruction can be computed efficiently. Finally, we illustrate the applicability of our approach on both decision trees and rule lists, by comparing the theoretical information leak associated to either exact or heuristic learning algorithms. Our results suggest that optimal interpretable models are often more compact and leak less information regarding their training data than greedily-built ones, for a given accuracy level.**

***Index Terms*—Reconstruction Attack, Privacy, Interpretability, Machine Learning**

I. Introduction

The growing deployment of machine learning systems in real-world decision making systems raises several ethical concerns. Among them, transparency is often pointed out as a fundamental requirement. On the one hand, some approaches consist in explaining *black-box* models, whose internals are either hidden or too complex to be understood by the user. Such *post-hoc explainability* techniques [2] include global explanations [3], [4] approximating the entire black-box and local explanations [5], [6] of individual predictions. One important drawback with these approaches is that they may be misleading by not reflecting the actual black-box model behavior, and can even be manipulated by malicious entities [7]–[11]. On the other hand, it is possible to learn inherently *interpretable models*, whose structure can be understood by human users. Common types of interpretable models [12] include rule lists [13], [14], rule sets [15] and decision trees [16].

Another key requirement for the deployment of machine learning models is privacy. Indeed, such models are often trained on large amounts of personal data, and it is necessary to ensure that they learn useful generic patterns without leaking private information about individuals. In this context, *inference attacks* [17]–[19] leverage the output of a computation (*e.g.*, a trained model) to retrieve information regarding its inputs (*e.g.*, a training dataset). Among other types of inferences attacks, *reconstruction attacks* [17]–[19], aim at reconstructing partly or entirely a model's training set.

While releasing interpretable models can be desirable from a transparency perspective, it intrinsically leaks information regarding the model's training data. For instance, previous work [1] exploited this information to build a *probabilistic* reconstruction of a decision tree's training set - effectively implementing a form of *reconstruction attack*. It is then possible to quantify the amount of information leaked by the model by measuring the uncertainty remaining within the reconstructed probabilistic dataset. Furthermore, while reconstructions are often not unique, the proposed representation is able to encode a whole set of possible ones. However, the approach relies on strong assumptions, such as statistical independence and uniform distribution of the random variables modeling the probabilistic dataset. While it allows probabilistic reconstructions from decision trees, it is not generic enough to encode more general types of knowledge, and cannot be used with other types of interpretable models, such as rule lists. In this work, we generalize the notion of probabilistic dataset by relaxing the aforementioned assumptions. In particular, we show how the success of such generalized probabilistic reconstructions can be assessed, and illustrate it theoretically and empirically on several forms of interpretable models. Our contributions can be summarized as follows:

- We generalize probabilistic datasets to represent any type of knowledge acquired from an interpretable model's structure.

- We extend the metric for quantifying the success of probabilistic reconstruction attacks in a more generic setting.
- Although computationally expensive in the generic setting, we show how the success of a probabilistic reconstruction attack may be decomposed under realistic assumptions regarding the structure of the interpretable model considered.
- We demonstrate that in the specific case of decision trees and rule lists, the success of a probabilistic reconstruction attack can be estimated efficiently, and theoretically compare the reconstruction quality between these two hypothesis classes.
- We implement the proposed approach and compare the reconstruction quality from optimal and heuristically-built models, for both decision trees and rule lists.

The outline of the paper is as follows. First in Section II, we review related works on reconstruction attacks before introducing the notion of probabilistic dataset reconstruction. In Section III, we generalize this notion to handle more generic forms of knowledge. We show in Section IV how the success of such generalized probabilistic reconstructions from interpretable models can be assessed. Finally, we illustrate the applicability of the approach by demonstrating how it can be used to compare the amount of information optimal models carry compared to greedily-built ones.

II. Related Work

In this section, we first provide a brief overview of the existing work with respect to reconstruction attacks. As this term has been used in the literature to encompass a wide variety of problems and approaches, we then focus on the state-of-the-art on probabilistic datasets reconstruction from interpretable models.

A. Reconstruction Attacks

Ensuring that the output of a computation over a dataset $\mathcal{V}$ cannot be used to retrieve private information about it is a fundamental objective in privacy [20]. In particular, inference attacks [17], [18] aim at retrieving information regarding $\mathcal{V}$ by only observing the outputs of the computation. In machine learning, such computation is usually a learning algorithm whose output is a trained model. Two distinct adversarial settings are generally considered [18], [19]. In the *black-box setting*, the adversary does not know the model's internals and can only query it through an API. In the *white-box setting*, the adversary has full knowledge of the model parameters. Diverse types of inference attacks have been proposed against machine learning models [19]. In this paper, we focus on *reconstruction attacks* [17]–[19], in which an adversary aims at recovering parts of a model's training data.

Reconstruction attacks have first been studied in the context of database access mechanisms. In this setup, a database contains records about individuals, with each record being composed of non-private information along with a private bit [17]. The adversary performs queries to a database access mechanism, whose outputs are aggregate and noisy statistics about private bits of individuals in the database. An efficient linear program for reconstructing private bits of the database leveraging counting queries was proposed [20] and later improved and generalized to handle other types of queries [21]. The practical effectiveness of such approaches was demonstrated in several real-world applications [22]–[24].

Other previous works have also tackled reconstruction problems in other settings. For example in the pharmacogenetics field, machine learning models predict medical treatments specific to a patient's genotype and background. In this sensitive context, a reconstruction attack was proposed, taking as input a trained model and some demographic (non-private) information about a patient whose records were used for training and predicting its sensitive attributes [25]. Subsequent works have designed reconstruction attacks leveraging confidence values output by machine learning models to infer private information about training examples given some information about them [26]. Other works have studied the intended [27] and unintended [28] training data memorization of machine learning models, along with different ways to exploit it in a white-box or black-box setting. In collaborative deep learning, it was also shown that an adversarial server can exploit the collected gradient updates to recover parts of the participants' data [29]. In addition, in the context of online learning, a reconstruction attack was developed to infer the *updating set* (*i.e.*, newly-collected data used to re-train the deployed model) information using a generative adversarial network leveraging the difference between the model before and after its update [30].

Finally, recent works have considered the special case of training set *sensitive attributes* reconstruction in the context of fair learning [31]–[34]. The key challenge here is that while *sensitive attributes* are usually known at training time to ensure the resulting model's fairness, they cannot be used explicitly for inference to avoid disparate treatment [35]. More precisely, [32] proposed a machine learning based attack leveraging an auxiliary dataset whose sensitive attributes are known, while [31], [34] explicitly exploit fairness by encoding it within declarative programming frameworks to enhance the reconstruction. Both [31] and [33] consider a particular setup in which a *learner* can query an *auditor* (owning the training set sensitive attributes) to known whether some model's parameters satisfy the fairness constraints. They show that the auditor's answers can be leveraged to conduct the attack.

To the best of our knowledge, the only attack leveraging the structure of a trained interpretable model to build a probabilistic reconstruction of its training set was proposed in [1]. Our *white-box* approach builds upon this baseline work whose key notions are introduced in details hereafter.

B. Probabilistic Dataset Reconstruction from Interpretable Models

In [1], the structure of a decision tree is used to build a probabilistic reconstruction of its training dataset, in the form of a *probabilistic dataset*, as introduced in Definition 1.

Definition 1: **(Probabilistic Dataset) [1].** A probabilistic dataset $\mathcal{V}$ is composed of n *examples* $\{x_1, \ldots, x_n\}$ (the dataset's rows), each consisting in a vector of d attributes

TABLE I: Example of deterministic dataset $\mathcal{V}^{Orig}$.

	a_1	a_2	a_3	Label
x_1	12	0	3	0
x_2	14	1	2	0
x_3	11	1	2	1
x_4	14	0	1	1

TABLE II: Example of probabilistic dataset $\mathcal{V}^{DT}$ reconstructed from a Decision Tree (Figure 1).

	a_1	a_2	a_3	Label
x_1	$\in \{12, 13, 14, 15\}$	$\in \{0, 1\}$	$\in \{2, 3\}$	0
x_2	$\in \{12, 13, 14, 15\}$	$\in \{0, 1\}$	$\in \{2, 3\}$	0
x_3	$\in \{10, 11\}$	$\in \{0, 1\}$	$\in \{2, 3\}$	1
x_4	$\in \{10, 11, 12, 13, 14, 15\}$	$\in \{0, 1\}$	$\in \{1\}$	1

$\{a_1, \ldots, a_d\}$ (the dataset's columns). Each attribute a_k has a domain of definition $\mathcal{V}_k$ that includes all the possible values of this attribute. The knowledge about attribute a_k of example x_i is modeled by a probability distribution over all the possible values of this attribute, using random variable $\mathcal{V}_{i,k}$. Importantly, variables $\{\mathcal{V}_{i\in[1..n],k\in[1...d]}\}$ are assumed to be statistically independent from each other and their probability distribution to be uniform.

In practice, if a particular value $v_{i,k} \in \mathcal{V}_k$ of an attribute gathers all the probability mass (*i.e.,* it is perfectly determined: $\mathbb{P}(\mathcal{V}_{i,k} = v_{i,k}) = 1$), then the attribute is said to be deterministic. By extension, a probabilistic dataset whose attributes are all deterministic (*i.e.*, the knowledge about the dataset is perfect) is called a *deterministic dataset*. Previous work [1] proposes a procedure to build a probabilistic dataset $\mathcal{V}^{DT}$ given the structure of a trained decision tree DT. Such probabilistic dataset gathers the knowledge that the decision tree inherently encodes about its (deterministic) training dataset $\mathcal{V}^{Orig}$. The construction of this probabilistic dataset can then be coined as a *probabilistic reconstruction attack*. By construction, $\mathcal{V}^{DT}$ is *compatible* with $\mathcal{V}^{Orig}$: the true value $v_{i,k}^{Orig}$ of any attribute a_k for any example x_i is always contained within the set of possible values for this attribute and this example in the probabilistic reconstruction: $\mathbb{P}\left(\mathcal{V}_{i,k}^{DT} = v_{i,k}^{Orig}\right) > 0$. A natural way to quantify the *success of the probabilistic reconstruction attack* is in terms of the average amount of uncertainty that remains in the built probabilistic dataset $\mathcal{V}^{DT}$, as stated in the following definition.

Definition 2: **(Measure of success of a probabilistic reconstruction attack) [1].** Let $\mathcal{V}^{Orig}$ be a deterministic dataset composed of n examples and d attributes, used to train a machine learning model M. Let $\mathcal{V}^M$ be a probabilistic dataset reconstructed from M. By construction, $\mathcal{V}^M$ is compatible with $\mathcal{V}^{Orig}$. The success of the reconstruction is quantified as the average uncertainty reduction over all attributes of all examples in the dataset:

$$\mathsf{Dist}(\mathcal{V}^M, \mathcal{V}^{Orig}) = \frac{1}{n \cdot d} \sum_{i=1}^{n} \sum_{k=1}^{d} \frac{H(\mathcal{V}_{i,k}^M)}{H(\mathcal{V}_{i,k})} \tag{1}$$

in which random variable $\mathcal{V}_{i,k}$ corresponds to an *uninformed* reconstruction, uniformly distributed over all possible values $\mathcal{V}_k$ of attribute a_k, and H denotes the Shannon entropy.

Smaller values of $\mathsf{Dist}(\mathcal{V}^M, \mathcal{V}^{Orig})$ indicate better reconstruction performance (*i.e.*, a more successful attack). In particular, if $\mathcal{V}^M = \mathcal{V}^{Orig}$, $\mathsf{Dist}(\mathcal{V}^M, \mathcal{V}^{Orig}) = 0$: the reconstruction is perfect and there is no uncertainty at all. In contrast, in other extreme in which $\mathcal{V}^M$ contains no knowledge at all, $\mathsf{Dist}(\mathcal{V}^M, \mathcal{V}^{Orig}) = 1$.

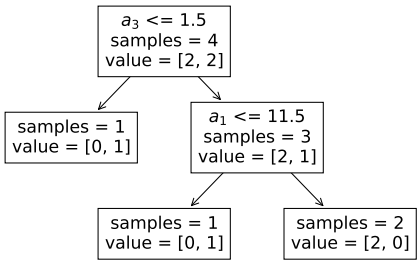


Fig. 1: Example of Decision Tree DT trained using `scikit-learn` [36], with 1.0 accuracy on $\mathcal{V}^{Orig}$ (Table I).

Remark 1: Definitions 1 and 2 are slightly more general than in [1]. Indeed, both use actual random variables while in [1] each attribute of each example is simply modeled via a set of possible values, which is only suitable under the assumed hypothesis of statistical independence and uniform distribution of the random variables. Thus, our extended formulation eases the generalization we further provide in Section III while encompassing this particular case.

We illustrate the reconstruction process proposed in [1] with a toy example. A deterministic dataset $\mathcal{V}^{Orig}$, provided in Table I is used to train a decision tree classifier DT depicted in Figure 1. This dataset includes four examples $x_{i\in\{1,2,3,4\}}$ with three attributes $a_{k\in\{1,2,3\}}$ with domains $\mathcal{V}_1 = \{10, 11, 12, 13, 14, 15\}$, $\mathcal{V}_2 = \{0, 1\}$ and $\mathcal{V}_3 = \{1, 2, 3\}$. This decision tree, learnt using the `scikit-learn` python library [36], provides the per-label number of training examples in each internal node and each leaf. Intuitively, its structure can then be used to reconstruct a probabilistic version of its training dataset $\mathcal{V}^{DT}$, given in Table II. The algorithm used to build $\mathcal{V}^{DT}$ simply follows each branch and performs the domains' reductions associated to each split along the branch.

Using Definition 2, we can compute the *success of the reconstruction* as the average amount of uncertainty contained within $\mathcal{V}^{DT}$. For instance, $\mathcal{V}_1 = \{10, 11, 12, 13, 14, 15\}$ and $\mathcal{V}_{3,1}^{DT}$ takes values in $\{10, 11\}$. Then, considering all the possible values as equally probable, the uncertainty reduction for attribute a_1 of example x_3 is: $\frac{H(\mathcal{V}_{3,1}^{DT})}{H(\mathcal{V}_{3,1})} = \frac{-log_2(\frac{1}{2})}{-log_2(\frac{1}{6})} \approx 0.387$. By averaging such computation over the entire dataset (*i.e.*, over all attributes of all examples), we obtain $\mathsf{Dist}(\mathcal{V}^{DT}, \mathcal{V}^{Orig}) \approx 0.736$. To facilitate reading, we aligned $\mathcal{V}^{Orig}$ and $\mathcal{V}^{DT}$. In practice, such alignment can be performed using the Hungarian

TABLE III: Example of deterministic dataset $\mathcal{V}^{Orig'}$.

	a'_1	a'_2	a'_3	Label
x'_1	1	1	1	1
x'_2	1	1	0	1
x'_3	0	1	1	0
x'_4	1	0	1	0
x'_5	1	0	0	1

algorithm [37], [38] as is done in [1]. In a nutshell, it consists in performing a minimum cost matching between the examples of $\mathcal{V}^{Orig}$ and those of $\mathcal{V}^{DT}$, where the assignment cost is computed as the sum of the distances between the paired examples. Intuitively, the objective is to determine *to which example within $\mathcal{V}^{Orig}$ corresponds each reconstructed example in $\mathcal{V}^{DT}$*. However, this would not be needed in scenarios in which $\mathcal{V}^{Orig}$ is unknown, as $\mathcal{V}^{DT}$ is compatible with $\mathcal{V}^{Orig}$ by construction, and Dist (1) does not require further information regarding $\mathcal{V}^{Orig}$.

Hereafter, we generalize the notions introduced in this section to be able to handle more general type of knowledge, corresponding to other types of interpretable models.

III. Generalizing Probabilistic Datasets Reconstruction

In this section, we first illustrate the limits of probabilistic datasets and motivate the need to relax some of their underlying assumptions. Consequently, we introduce *generalized probabilistic datasets*, which can be used to encode any arbitrary knowledge regarding a dataset. Finally, we define a generalized metric Dist_G that can be used to quantify uncertainty reduction within such datasets.

A. Motivation

The concept of probabilistic dataset as described in Definition 1 is suitable to encode knowledge regarding a dataset, as long as this knowledge involves each *cell* (*i.e.*, which corresponds to one attribute for one example) individually. For instance, this is appropriate for decision trees in which an example is classified *exactly* by one branch. Furthermore, each branch corresponds to a *conjunction* (*i.e.*, logical AND) of conditions (*splits*) over features, which all have to be satisfied. These conditions allow for the reduction of each such feature's domains *individually*. However, for other representations of interpretable classifiers, such as rule lists or rule sets, this condition will not be valid. Again, we illustrate this observation using a toy example.

More precisely, Rule List 1 was trained on (deterministic) dataset $\mathcal{V}^{Orig'}$, shown in Table III. It gathers five examples $x'_{i\in\{1,2,3,4,5\}}$ described by three binary attributes, named $a'_{k\in\{1,2,3\}}$ (with domains $\mathcal{V}'_{k\in\{1,2,3\}} = \{0,1\}$). For each rule (including the default rule), RL indicates the number of training examples it captures, for each class. For example, the second rule captures two training examples belonging to class 0 (here, x'_3 and x'_4).

TABLE IV: Example of (generalized) probabilistic dataset $\mathcal{W}^{RL}$ reconstructed from Rule List 1.

	a'_1	a'_2	a'_3	Label
x'_1	1	1	$\in\{0,1\}$	1
x'_2	1	1	$\in\{0,1\}$	1
x'_3	$\in\{(0,0),(0,1),(1,0)\}$		1	0
x'_4	$\in\{(0,0),(0,1),(1,0)\}$		1	0
x'_5	$\in\{(0,0),(0,1),(1,0)\}$		0	1

```
if [a'_1] and [a'_2] then [true] ([0 ; 2] examples)
else if [a'_3] then [false] ([2 ; 0] example)
else [true] ([0 ; 1] example)
```

Rule list 1: Example rule list RL trained using COREL S [14], [39], with 1.0 accuracy on $\mathcal{V}^{Orig'}$ (Table III).

The algorithm reconstructing a probabilistic version of a rule list RL's training set from RL itself simply follows the path of each example. For an example classified by the jth rule, it reduces the domains of the attributes involved in the jth rule accordingly. It also eliminates all attributes' conjunctions contradicting the fact that the example did not match the previous rules within RL.

For instance, the following knowledge can be extracted from Rule List 1:

- The **first rule** indicates that for 2 (positive) examples, the two Boolean attributes a'_1 and a'_2 are true.
- Using the **second rule**, we know that the Boolean attribute a'_3 is true for 2 (negatively-labelled) examples. *Furthermore, we know that a'_1 and a'_2 can not be simultaneously true for these examples (or else they would have been captured by the first rule).*
- Finally, the **default rule** states that for 1 (positively-labelled) example, a'_3 is false, and a'_1 *and* a'_2 *can not be simultaneously true*.

Using such knowledge, one can build a (generalized) probabilistic dataset as shown in Table IV. In this example, part of the model's knowledge directly reduces the individual domains of some attributes for the concerned examples. As such, the information it brings will successfully be quantified by Dist and encoded in a probabilistic dataset. However, other information (specified *in italic*) does not reduce any attribute's domain individually. For instance, as shown in Table IV, one knows that for examples x'_3 and x'_4, a'_1 and a'_2 can not simultaneously be true. Nevertheless, taken apart, their respective domains would be unchanged as both binary attributes can still take values in $\{0,1\}$. While such knowledge brings information for reconstruction, this cannot be quantified using Dist nor represented using a probabilistic dataset as formalized in Definition 1.

Indeed, one key assumption with Definition 1 is that the random variables representing each attribute for each example are independent from each other. This is leveraged by Dist, which computes the reductions of the individual entropies. However, this representation cannot handle more generic knowledge, in which uncertainty can be spread jointly across multiple random variables. This limitation is also pointed

out in the theory of probabilistic databases. More precisely, quoting [40], this representation (talking about a scheme similar to probabilistic datasets as formalized in Definition 1 and illustrated in Figure II) is "more compact", as we do not need to expand all possible combinations of the different variables' values explicitly. However, "it cannot account for correlations across possible readings of different fields, such as when we know that no two persons can have the same social security number". In this particular case, this corresponds to a correlation across examples, while in the aforementioned example of Rule List 1 we observed correlations between attributes within the same example. For instance, to encode the knowledge regarding attributes a'_1 and a'_2 of examples x'_3 and x'_4, we had to enumerate all the possible combinations of these two attributes' values (Table IV).

B. Generalized Probabilistic Datasets

As illustrated in the previous subsection, the assumptions underlying probabilistic datasets (Definition 1) - namely statistical independence and uniform distribution of their random variables - make them inappropriate in the general case. Generalized probabilistic datasets remove these assumptions as stated in Definition 3.

Definition 3: **(Generalized probabilistic dataset).** A generalized probabilistic dataset $\mathcal{W}$ is composed of n *examples* $\{x_1, \ldots, x_n\}$ (the dataset's rows), each consisting in a vector of d attributes $\{a_1, \ldots, a_d\}$ (the dataset's columns). The knowledge about attribute a_k of example x_i is modeled by a probability distribution over all the possible values of this attribute, using random variable $\mathcal{W}_{i,k}$. Importantly, variables $\{\mathcal{W}_{i\in[1..n],k\in[1..d]}\}$ are not necessarily statistically independent from each other and can follow any arbitrary distribution. Each possible instantiation $w = \{w_{i\in[1..n],k\in[1..d]}\}$ of the $\mathcal{W}_{i\in[1..n],k\in[1..d]}$ variables (*i.e.*, each deterministic dataset compatible with $\mathcal{W}$) is named a *possible world*. We let $\Pi(\mathcal{W})$ denote the set of possible worlds within $\mathcal{W}$: $\Pi(\mathcal{W}) = \{w \mid \mathbb{P}(\mathcal{W}_{i\in[1..n],k\in[1..d]} = w_{i\in[1..n],k\in[1..d]}) > 0\}$.

Again, if all its variables are determined, a generalized probabilistic dataset is said to deterministic. A key difference between probabilistic datasets and their generalized counterparts is that the set of possible worlds of a probabilistic dataset simply consists in all combinations of the possible variables' values, all random variables being statistically independent. For generalized probabilistic datasets, it is not the case as there can exist complex inter-dependencies between the random variables that directly influence $\Pi(\mathcal{W})$ (as illustrated in Section III-A). Importantly, note that generalized probabilistic datasets as introduced in Definition 3 can encode any knowledge regarding the inferred reconstruction, as they simply consist in a set of random variables $\{\mathcal{W}_{i\in[1..n],k\in[1..d]}\}$ with no further assumption regarding their distribution. Furthermore, because these variables can be categorical, but also continuous, there is no particular restriction regarding the type of attributes contained in the reconstructed dataset.

Our generalized probabilistic dataset definition matches the notions of *probabilistic* or *incomplete databases* that are used in the theory of probabilistic databases [40]. Both correspond to database representations in which some values are uncertain. More precisely, an *incomplete database* defines a set of *possible worlds*, denoting the possible states of the database (*i.e.*, set of values for the different relations). Such *worlds* are also called *relational database instances*, and correspond to all deterministic datasets compatible with our generalized probabilistic dataset in the context of this work. If one can associate a probability to each possible world, then the database is called a *probabilistic database* - which generalizes incomplete databases. In the context of this work, one could leverage external knowledge (*e.g.*, demographic information about the data distribution) to associate probabilities to the possible worlds in $\Pi(\mathcal{W})$. This would lead to a reduction of the uncertainty of the dataset (thus lowering its joint entropy and raising the reconstruction success).

Both incomplete and probabilistic databases are semantic definitions for which designing a practical representation is challenging [40]. Indeed, it is always possible to represent a probabilistic database (here, a generalized probabilistic dataset $\mathcal{W}$) by explicitly enumerating all its possible worlds (here, $\Pi(\mathcal{W})$). However, such a mechanism is difficultly applicable when the set of possible worlds is very large. To circumvent this issue, some *compact* representations have been proposed. For instance, in *conditional tables* (or *c-tables*), the different values of the database cells are associated to a propositional formula, called condition, over some random variables. The different assignments of the random variables define the different states of the database (*i.e.*, possible worlds). *Probabilistic conditional tables* (or *pc-tables*) extend this concept by assigning probabilities to the conditional variables assignments. While (p)c-tables may be an interesting representation for generalized probabilistic datasets, we do not assume any specific representation for our generalized probabilistic datasets in this work. Rather, we demonstrate in Section IV that in the context of training set reconstruction from an interpretable model, we can quantify the amount of uncertainty that remains in the resulting generalized probabilistic dataset without building it explicitly (which in practice may be infeasible even with efficient structures such as c-tables).

C. Generalized Measure of the Attack Success

We now generalize the metric introduced in Definition 2 to quantify the success of a probabilistic reconstruction attack. As stated in Definition 4, our new metric Dist_G is more general as it quantifies the uncertainty reduction on the entire dataset using the joint entropy of the underlying random variables.

Definition 4: **(Generalized measure of success of a probabilistic reconstruction attack).** Let $\mathcal{W}^{Orig}$ be a deterministic dataset composed of n examples and d attributes, used to train a machine learning model M. Let $\mathcal{W}^M$ be a generalized probabilistic dataset reconstructed from M. By construction, $\mathcal{W}^M$ is compatible with $\mathcal{W}^{Orig}$ (*i.e.*, $\mathcal{W}^{Orig} \in \Pi(\mathcal{W}^M)$). The success of the performed reconstruction is quantified as the

overall uncertainty reduction in the dataset:

$$\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig}) = \frac{H\left(\{\mathcal{W}^M_{i,k} \mid i \in [1..n], k \in [1..d]\}\right)}{H\left(\{\mathcal{W}_{i,k} \mid i \in [1..n], k \in [1..d]\}\right)} \tag{2}$$

$$= \frac{\sum_{w \in \Pi(\mathcal{W}^M)} -\mathbb{P}(w) \cdot log_2(\mathbb{P}(w))}{\sum_{i=1}^{n} \sum_{k=1}^{d} H(\mathcal{W}_{i,k})} \tag{3}$$

in which H denotes the Shannon entropy (or joint entropy, when applied to a set of variables, as in (2)), and random variable $\mathcal{W}_{i,k}$ corresponds to an *uninformed* reconstruction, uniformly distributed over all possible values of attribute a_k.

The denominator in Equation (2) can be decomposed as a sum in Equation (3) because the random variables $\mathcal{W}_{i \in [1..n], k \in [1..d]}$ are independent from each other, and the joint entropy of a set of variables is equal to the sum of the individual entropies of the variables in the set if and only if the variables are statistically independent. This is not the case for variables $\mathcal{W}^M_{i \in [1..n], k \in [1..d]}$, and thus the generalized probabilistic dataset has to be considered as a whole through its set of possible worlds $\Pi(\mathcal{W}^M)$. Again, note that Definition 4 is general enough to quantify the uncertainty reduction brought by any type of knowledge, as no assumption is made regarding the distribution of the generalized probabilistic dataset variables, and Dist_G considers their joint entropy. Furthermore, to handle continuous attributes, the sum over the set of possible worlds $\Pi(\mathcal{W}^M)$ in Equation (3) can be changed into an integral calculation.

The key properties of Dist also hold for Dist_G. In particular, for any deterministic dataset $\mathcal{W}^{Orig}$, we have $\mathsf{Dist}_G(\mathcal{W}^{Orig}, \mathcal{W}^{Orig}) = 0$. Furthermore, if $\mathcal{W}^M$ contains no knowledge at all, we have that $\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig}) = 1$ for any deterministic dataset $\mathcal{W}^{Orig}$.

One important difference between Dist and Dist_G is the fact, that due to its averaging over the per-example-per-attribute individual uncertainty reductions, Dist considers all features equal (in terms of contribution to the overall uncertainty) while it is not the case for Dist_G. To illustrate this, let us assume a toy scenario with a (deterministic) dataset $\mathcal{V}^{Orig}$ with a single record $x_1 = (1, 1)$ and two attributes a_1 and a_2 with domains $\mathcal{V}_1 = \{0, 1\}$ and $\mathcal{V}_2 = \{1, 2, 3\}$. Consider the two probabilistic datasets $\mathcal{V}^{rec1}$, in which we know that for x_1, $a_1 = 1$, and $\mathcal{V}^{rec2}$, in which we know that for x_1, $a_2 = 1$. These datasets are summarized in Tables V, VI and VII.

TABLE V: $\mathcal{V}^{Orig}$

	a_1	a_2
x_1	1	1

TABLE VI: $\mathcal{V}^{rec1}$

	a_1	a_2
x_1	1	$\in \{1, 2, 3\}$

TABLE VII: $\mathcal{V}^{rec2}$

	a_1	a_2
x_1	$\in \{0, 1\}$	1

Using Definition 2, we have: $\mathsf{Dist}(\mathcal{V}^{rec1}, \mathcal{V}^{Orig}) = 0.5$, as:

$$\frac{H(\mathcal{V}^{rec1}_{1,1})}{H(\mathcal{V}_{1,1})} = \frac{-log_2(1)}{-log_2(\frac{1}{2})} = 0 \quad \text{and} \quad \frac{H(\mathcal{V}^{rec1}_{1,2})}{H(\mathcal{V}_{1,2})} = \frac{-log_2(\frac{1}{3})}{-log_2(\frac{1}{3})} = 1.$$

Conversely, we also have $\mathsf{Dist}(\mathcal{V}^{rec2}, \mathcal{V}^{Orig}) = 0.5$ because:

$$\frac{H(\mathcal{V}^{rec2}_{1,1})}{H(\mathcal{V}_{1,1})} = \frac{-log_2(\frac{1}{2})}{-log_2(\frac{1}{2})} = 1 \text{ and } \frac{H(\mathcal{V}^{rec2}_{1,2})}{H(\mathcal{V}_{1,2})} = \frac{-log_2(1)}{-log_2(\frac{1}{3})} = 0.$$

However, out of 6 possible reconstructions for x_1 (without any knowledge), 3 are possible within $\mathcal{V}^{rec1}$ while only 2 are possible with $\mathcal{V}^{rec2}$. Intuitively, $\mathcal{V}^{rec2}$ yields more information (or, conversely, less uncertainty) than $\mathcal{V}^{rec1}$, but Dist cannot account for this difference due to normalization and individual measure of entropy across examples' attributes. For notation consistency, we associate to these datasets their generalized counterparts $\mathcal{W}^{Orig}$, $\mathcal{W}^{rec1}$ and $\mathcal{W}^{rec2}$, containing the exact same information (recall that probabilistic datasets are simply a particular case of generalized probabilistic datasets, in which the dataset's variables are statistically independent and uniformly distributed). Using our generalized metric introduced in Definition 4, we have:

$$\mathsf{Dist}_G(\mathcal{W}^{rec1}, \mathcal{W}^{Orig}) = \frac{H\left(\{\mathcal{W}^{rec1}_{1,1}, \mathcal{W}^{rec1}_{1,2}\}\right)}{H\left(\{\mathcal{W}_{1,1}, \mathcal{W}_{1,2}\}\right)} = \frac{-log_2(\frac{1}{3})}{-log_2(\frac{1}{6})} \approx 0.613$$

$$\text{and} \quad \mathsf{Dist}_G(\mathcal{W}^{rec2}, \mathcal{W}^{Orig}) = \frac{H\left(\{\mathcal{W}^{rec2}_{1,1}, \mathcal{W}^{rec2}_{1,2}\}\right)}{H\left(\{\mathcal{W}_{1,1}, \mathcal{W}_{1,2}\}\right)} = \frac{-log_2(\frac{1}{2})}{-log_2(\frac{1}{6})} \approx 0.387$$

As lower values indicate less uncertainty (*i.e.*, better reconstruction performances), we observe that Dist_G successfully distinguishes between $\mathcal{W}^{rec1}$ and $\mathcal{W}^{rec2}$. Thus by avoiding the drawbacks of the normalization across dataset cells, the new metric Dist_G successfully takes into account the specificities of the two probabilistic datasets.

IV. Generalized Probabilistic Datasets Reconstruction from Interpretable Models

We now investigate how to quantify the success of a probabilistic reconstruction attack in practice. First, we discuss how the attack success computation can be decomposed under reasonable assumptions regarding the structure of the interpretable model considered. Then, we show how it can be computed without explicitly building the entire set of possible worlds, as long as one is able to count them. Finally, we demonstrate that such simplification is possible for decision trees as well as rule lists models, and theoretically compare the reconstruction quality from these two hypothesis classes.

A. General Case

Let $\mathcal{W}^M$ be a generalized probabilistic dataset reconstructed from an interpretable model M. As stated in Definition 4, the success of the probabilistic reconstruction attack can be quantified using Dist_G. One can observe that the denominator $\left(\sum_{i=1}^{n} \sum_{k=1}^{d} H(\mathcal{W}_{i,k})\right)$ is a constant, only depending on the attributes' domains $\mathcal{V}_{k \in [1..d]}$. Indeed, variables $\mathcal{W}_{i \in [1..n], k}$ are uniformly distributed over $\mathcal{V}_k$ (the domain of attribute a_k) and so $H(\mathcal{W}_{i,k}) = -log_2\left(\frac{1}{|\mathcal{V}_k|}\right)$. Thus:

$$\sum_{i=1}^{n} \sum_{k=1}^{d} H(\mathcal{W}_{i,k}) = n \cdot \sum_{k=1}^{d} -log_2\left(\frac{1}{|\mathcal{V}_k|}\right) \tag{4}$$

As the denominator in Equation (2) is a constant that can be easily computed, we will focus only on the numerator in the remaining of this section, using the following notation:

$$\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig}) \propto H\left(\{\mathcal{W}^M_{i\in[1..n],k\in[1..d]}\}\right) \quad (5)$$

1) Independence assumptions: decomposing the attack success computation.: In the general case, the computation of the joint entropy of the generalized probabilistic dataset's cells must be done through its set of possible worlds $\Pi(\mathcal{W}^M)$, as shown in Equation (3). However, if one can establish the statistical independence of some of the $\mathcal{W}^M_{i,k}$ variables, this computation can be further decomposed. Indeed, the joint entropy of a set of statistically independent variables is equal to the sum of their individual entropies. For instance, if the knowledge of model M applies to each example $x_{i\in[1..n]}$ independently, the sets of variables $\{\mathcal{W}^M_{i,k\in[1..d]}\}_{i\in[1..n]}$ are independent from each other. This condition is satisfied if M is a decision tree or a rule list, because each example is captured by exactly one "decision path" (*i.e.*, branch or rule). Indeed, this decision path reduces the set of possible reconstructions for each example x_i independently from the other examples. By a slight abuse of notation, we let $\Pi_i(\mathcal{W}^M)$ denote the set of possible worlds (*i.e.*, reconstructions) for example (row) x_i. As a consequence, we have:

$$\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig}) \propto \sum_{i=1}^{n} H\left(\{\mathcal{W}^M_{i,k\in[1..d]}\}\right) \quad (6)$$

$$\propto \sum_{i=1}^{n}\left(\sum_{w_i\in\Pi_i(\mathcal{W}^M)} -\mathbb{P}(w_i)\cdot log_2(\mathbb{P}(w_i))\right) \quad (7)$$

While Equation (7) holds for both rule lists and decision trees, its computation can be further decomposed for the later. Indeed, in a decision tree each example is classified by exactly one branch, and such branch defines a conjunction of Boolean conditions over attributes' values, called *splits*. Such conditions must all be satisfied for the example to be captured by the branch - hence all the concerned attributes' domains can be reduced individually. As a consequence, this implies that all variables $\mathcal{W}^M_{i\in[1..n],k\in[1..d]}$ are actually statistically independent resulting in:

$$\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig}) \propto \sum_{i=1}^{n}\sum_{k=1}^{d} H\left(\mathcal{W}^M_{i,k}\right) \quad (8)$$

Note that Equation (8) corresponds to the particular case studied in [1], with the computation being exactly as for their proposed Dist metric (Definition 2), with the only difference being the absence of normalization. Observe that Equation (8) does not hold in general for rule list models due to the fact that for a given example, the information that it did not match previous rules within the rule list corresponds to negating a conjunction, hence producing a disjunction. As a result, this potentially breaks the statistical independence between some of the $\{\mathcal{W}^M_{i,k\in[1..d]}\}$ variables.

2) Uniform distribution assumptions: efficient attack success computation.: The explicit enumeration of the possible worlds $\Pi(\mathcal{W}^M)$ is not practically conceivable for real-size datasets. However, quantifying a probabilistic reconstruction attack success can sometimes be done only by computing their number $|\Pi(\mathcal{W}^M)|$. Indeed, assuming a uniform probability distribution between them, one can then easily quantify the amount of uncertainty using Dist_G (Definition 4), as $\forall w \in \Pi(\mathcal{W}^M), \mathbb{P}(w) = \frac{1}{|\Pi(\mathcal{W}^M)|}$, resulting in:

$$\sum_{w\in\Pi(\mathcal{W}^M)} -\mathbb{P}(w)\cdot log_2(\mathbb{P}(w)) = -log_2\left(\frac{1}{|\Pi(\mathcal{W}^M)|}\right) \quad (9)$$

Remark that only the number of possible worlds $|\Pi(\mathcal{W}^M)|$ is needed to compute Equation (9). In the general case, this number cannot be retrieved without building $\Pi(\mathcal{W}^M)$ explicitly. However, several types of interpretable models enable to compute $|\Pi(\mathcal{W}^M)|$ efficiently (*i.e.*, without building $\Pi(\mathcal{W}^M)$). For instance, this is the case when reconstructing generalized probabilistic datasets from decision tree or rule list models. Plugging together Equations (7) and (9), we have:

$$\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig}) \propto \sum_{i=1}^{n} -log_2\left(\frac{1}{|\Pi_i(\mathcal{W}^M)|}\right) \quad (10)$$

In the next subsections, we demonstrate how $|\Pi_i(\mathcal{W}^M)|_{i\in[1..n]}$ can be computed in polynomial time (with respect to the model's size) for decision trees and rule lists. Note that the assumptions performed in this subsection are realistic. In particular, the uniform distribution assumption simply means that if the model at hand is compatible with several reconstructions (deterministic datasets), they are all equally likely (one can not state whether one is more likely than the others). Furthermore, the independence assumption (between the dataset rows, as in (6)) is mainly challenging when dealing with ensemble models (and matching the knowledge of the different base learners is an additional challenge).

B. Decision Trees

Let DT be a decision tree with r branches, in which each branch $f_{j\in[1..r]}$ is a conjunction of Boolean assertions over attributes' values ending with a leaf prediction. The value $\mathsf{num}(f_j)$ represents the number of different examples (*i.e.*, number of different combinations of attributes values) that satisfy f_j. It can be computed by multiplying the cardinalities of the reduced domains. Thus, for each example x_i classified by branch f_j, we have $|\Pi_i(\mathcal{W}^{DT})| = \mathsf{num}(f_j)$. Additionally, $C_{j\in[1..r]}$ is defined as the support of each leaf (*i.e.*, the number of training examples captured by the leaf, as indicated in the decision tree of Figure 1). Importantly, the tree branches partition the set of examples (as the leaves' supports are all disjoints), so we have $\sum_{j\in[1..r]} C_j = n$. Furthermore, the sum of Equation (10) which was performed over all n examples can be replaced with a sum over the r branches, with the entropy of each branch f_j being weighted by its support C_j. Plugging these new notions into Equation (10), we obtain that the overall

joint entropy of the reconstructed probabilistic version of DT's training set is:

$$\mathsf{Dist}_G(\mathcal{W}^{DT}, \mathcal{W}^{Orig}) \propto \sum_{j=1}^{r} -C_j \cdot log_2\left(\frac{1}{\mathsf{num}(f_j)}\right) \quad (11)$$

C. Rule Lists

Let $RL = (f_1, v_1)\dots(f_{r'}, v_{r'})$ be a rule list, following the notation introduced in [13]. Each term $f_{j\in[1..r']}$ is a conjunction of Boolean assertions over attributes' values and $v_{j\in[1..r']}$ is a prediction. Rule r' is the default decision, with $f_{r'}$ being the constant value True. Similarly to the leaves of a decision tree, each rule j is associated with its support C_j. Again, let $\mathsf{num}(f_j)$ denote the number of different examples (*i.e.*, number of different combinations of attributes values) that satisfy f_j. As a branch, a rule corresponds to a conjunction, hence $\mathsf{num}(f_j)$ can be computed easily by simply multiplying the cardinalities of the attributes' reduced domains.

Finally, we define $\forall 1 \leq j \leq r',\ \mathsf{Capt}_{RL}(f_j)$ as the number of possible different examples (*i.e.*, number of different combinations of attributes values) that f_j captures *within* RL (*i.e.*, examples satisfying f_j while not matching the antecedents of the previous rules within RL). As a particular case, note that we always have $\mathsf{Capt}_{RL}(f_1) = \mathsf{num}(f_1)$ as the first rule of any rule list is always applied first. For $1 \leq j \leq r'$, a straightforward general formulation is:

$$\mathsf{Capt}_{RL}(f_j) = \mathsf{num}(f_j \wedge \bigwedge_{l=1}^{j-1} \neg f_l) \quad (12)$$

The main challenge is that $\mathsf{num}(\bigwedge_{l=1}^{j-1} \neg f_l)$, in which $\bigwedge_{l=1}^{j-1} \neg f_l$ is the conjunction of the negations of the previous rules' antecedents, cannot be computed directly as $f_{l\in[1..j-1]}$ may overlap. Indeed, each antecedent f_l is a conjunction - hence its negation is a disjunction. More precisely, overall we get a conjunction of disjunctions, which means that the number of possible examples it characterizes cannot be computed by simply multiplying attributes' cardinalities as the different disjunctions may overlap. By a slight abuse of notation, we define for $1 \leq l \leq j \leq r',\ \mathsf{Capt}_{RL}(f_l, f_j)$ as the number of possible different examples (*i.e.*, number of different combinations of features values) that f_j could capture but that are actually captured by f_l in RL:

$$\mathsf{Capt}_{RL}(f_l, f_j) = \mathsf{num}(f_l \wedge f_j) - \sum_{h=1}^{l-1} \mathsf{Capt}_{RL}(f_h, (f_l \wedge f_j)) \quad (13)$$

The first term corresponds to the overlap between f_l and f_j, while the second one subtracts the unique examples within this overlap that are actually captured by rules placed before f_l in RL. Then:

$$\mathsf{Capt}_{RL}(f_j) = \mathsf{Capt}_{RL}(f_j, f_j) \quad (14)$$

$$= \mathsf{num}(f_j) - \sum_{l=1}^{j-1} \mathsf{Capt}_{RL}(f_l, f_j) \quad (15)$$

Just like the branches of a decision tree, the rules within a rule list partition the set of examples (as each example is captured by exactly one rule in the rule list). Then, the sum over all n examples in Equation (10) can be reformulated using a sum over the r' rules, with each rule's entropy being weighted by its support. Then, plugging (15) into (10), we obtain:

$$\mathsf{Dist}_G(\mathcal{W}^{RL}, \mathcal{W}^{Orig}) \propto \sum_{j=1}^{r'} -C_j \cdot log_2\left(\frac{1}{\mathsf{num}(f_j) - \sum_{l=1}^{j-1} \mathsf{Capt}_{RL}(f_l, f_j)}\right) \quad (16)$$

Comparing Decision Trees and Rule Lists: Comparing (16) to (11), we observe that an additional term is subtracted to the denominator of (16). This term corresponds to the information that the examples captured by rule j did not match any of the previous rules $l < j$ within RL. By lowering the denominator, it raises the overall success of the probabilistic reconstruction attack. There is no such term in (11) because there can be no overlap between a decision tree's leaves' supports. On the contrary, the rules within a rule list can overlap because they are ordered. Overall, these theoretical results confirm that rule lists are more expressive than decision trees, encoding more information than a decision tree of equivalent size [13]. The reconstruction methods depicted in Sections II-B and III-A have a computational cost linear in the models' sizes, as they simply follow each decision path once. However, the computation is more expensive for rule lists if one wants to avoid explicitly building the set of possible reconstructions, and the computational cost of (16) is rather factorial in the number of rules (which can be proved by induction). This suggests that the additional information brought by rule lists is also more challenging to extract.

V. Experiments

While our proposed metric precisely quantifies the amount of information an interpretable model carries regarding its training dataset, the aim of this section is to illustrate its practical usefulness through an example use. More precisely, we will investigate the differences between optimal and heuristically-built models, for both rule lists and decision trees.

A. Setup

In these experiments, we use both optimal and heuristic learning algorithms to compute decision trees and rule lists of varied sizes. Furthermore, optimal models are learnt optimizing solely accuracy, to avoid interference with other regularization terms. All details regarding the considered experimental setup are provided hereafter.

a) Learning algorithms: We use the following learning algorithms:

- **Optimal decision trees.** We use the `DL8.5` algorithm [41], [42] through its Python binding[1].

[1] https://github.com/aia-uclouvain/pydl8.5

- **Heuristic decision trees.** We use an optimized version of the CART greedy algorithm [16], as implemented within the scikit-learn[2] Python library [36] with its DecisionTreeClassifier object. We coin this method sklearn_DT.
- **Optimal rule lists.** We use the CORELS algorithm [14], [39] through its Python binding[3].
- **Heuristic rule lists.** While some implementations exist in the literature for building *heuristic rule lists* (for example, one is provided within the imodels[4] library[5] [43]), they do not offer precise control over the desired rule support and/or maximum rule list depth. For this reason, we implemented a CART-like greedy algorithm (close to the imodels' implementation), that we coin GreedyRL. In a nutshell, this algorithm selects the rule yielding to the best Gini impurity improvement at each level of the rule list, in a top-down manner.

b) Datasets: We use two datasets (binarized, as required by CORELS) which are very popular in the trustworthy machine learning literature. First, the UCI Adult Income dataset[6] [44] contains data regarding the 1994 U.S. census, with the objective of predicting whether a person earns more than $50K/year. Numerical features are discretized using quantiles and categorical features are one-hot encoded. The resulting dataset includes $48,842$ examples and 24 binary features. As DL8.5 was unable to learn optimal models within the specified time and memory limits for the largest size constraints, we randomly sub-sample 10% of the whole dataset. Second, the COMPAS dataset (analyzed by [45]) gathers records about criminal offenders in the Broward County of Florida collected from 2013 and 2014, with the task being recidivism prediction. We consider its discretized version used to evaluate CORELS in [14], consisting in $7,214$ examples characterized with 27 binary features[7].

c) Experimental Parameters: For each experiment, we randomly select 80% of the dataset to form a training set, and use the remaining 20% as a test set to ensure that models generalize well. We repeat the experiment five times using different seeds for the random train/test split, and report results averaged across the five runs. All experiments are run on a computing cluster over a set of homogeneous nodes using Intel Platinum 8260 Cascade Lake @ 2.4Ghz CPU. Each training phase is limited to one hour of CPU time and 12 GB of RAM. Within the proposed experimental setup, all models produced by the optimal learning algorithms (DL8.5 for decision trees or CORELS for rule lists) are certifiably optimal.

d) Models Learning: We set various constraints to the decision tree building algorithms, using maximum tree depths between 1 and 10 (ranging linearly by steps of 1) and (relative) minimum leaf supports between 0.01 and 0.05 (ranging linearly by steps of 0.01). For the rule list learning algorithms, we proceed identically and generate rule lists with various size constraints, using maximum depths *(number of rules within the rule list)* between 1 and 10 (ranging linearly by steps of 1) and (relative) minimum rule supports between 0.01 and 0.05 (ranging linearly by steps of 0.01). As we are interested in the optimality guarantee, we consider rules consisting in a single binary attribute (or its negation). Indeed, in our experiments, CORELS was unable to reach and certify optimality while also considering conjunction of features, as it dramatically increases the number of rules - and consequently, the algorithm search space. Finally, we set CORELS's sparsity regularization coefficient to a value small enough (*i.e.*, smaller than $\frac{1}{n}$) to ensure that only accuracy is optimized. All methods' parameters are left to their default value.

e) Resources: All scripts needed to reproduce our experiments, as well as our implementation of the greedy rule list learning algorithm GreedyRL are publicly available[8].

B. Results

After having learnt optimal and heuristic decision trees and rule lists under various constraints, we compute the amount of information they contain regarding their training sets using Dist_G, leveraging the computational tricks presented in Equations (11) and (16). As discussed in Section IV, we make the uniform distribution assumption, because the sole structure of a decision tree or rule list does not allow distinguishing between different compatible reconstructions. We also make the independence assumption between the different examples, because in both decision trees and rule lists, each example is captured by a single decision path (*i.e.,* leaf or rule). However, we do not make the independence assumption between the different attributes of each example, because as illustrated in Section III-A, it does not hold for rule lists models. Recall that lower uncertainty values indicate better reconstruction performances. We relate this value to two dimensions: the sizes of the models and their training accuracy. The former corresponds respectively to the number of splits performed in a decision tree or to the number of rules for a width-1 rule list. The later indicates the model's performance on its training set - *i.e.*, exactly what we aim at optimizing.

Results are provided for our experiments comparing exact and greedily-built decision trees and rule lists respectively in Figures 2 and 3. We observe the same trends for the two types of models. First, one can observe in Figures 2a and 3a that optimal models usually represent more information in a more compact way: the reconstruction uncertainty decreases faster for optimal models than with greedily-built ones. However, while for a given size optimal models contain more information regarding their training data, they are also way more accurate. This dimension is observed in Figures 2b and 3b. More precisely, we consistently observe that for a given accuracy level, optimal models always leak less information regarding

[2]https://scikit-learn.org/

[3]https://github.com/corels/pycorels

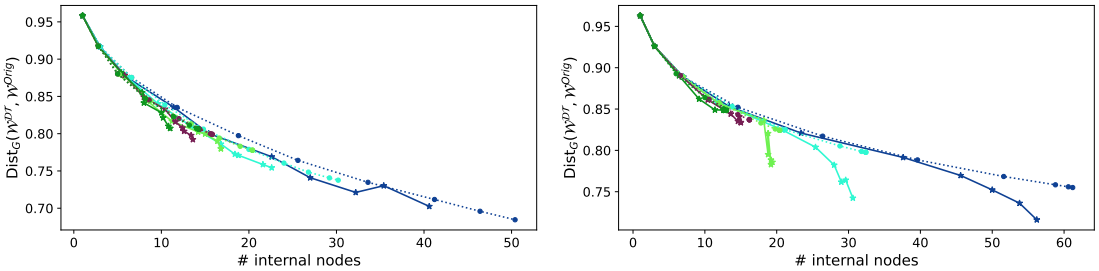
[4]https://github.com/csinva/imodels

[5]imodels is a Python library gathering tools to learn different types of popular interpretable machine learning models.

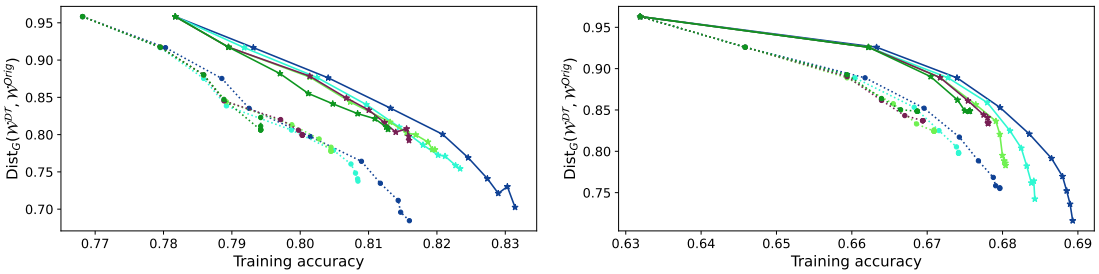
[6]https://archive.ics.uci.edu/ml/datasets/adult

[7]https://github.com/corels/pycorels/blob/master/examples/data/compas.csv

[8]https://github.com/ferryjul/ProbabilisticDatasetsReconstruction



(a) Entropy reduction as a function of the tree size (number of splits/internal nodes). Note that the number of internal nodes is restricted by the maximum depth constraint, but also by the enforced minimum leaf support (as mentioned in section V-A), which explains why the largest trees only have slightly more than 60 nodes.



(b) Entropy reduction as a function of training accuracy.

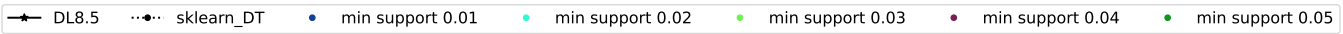


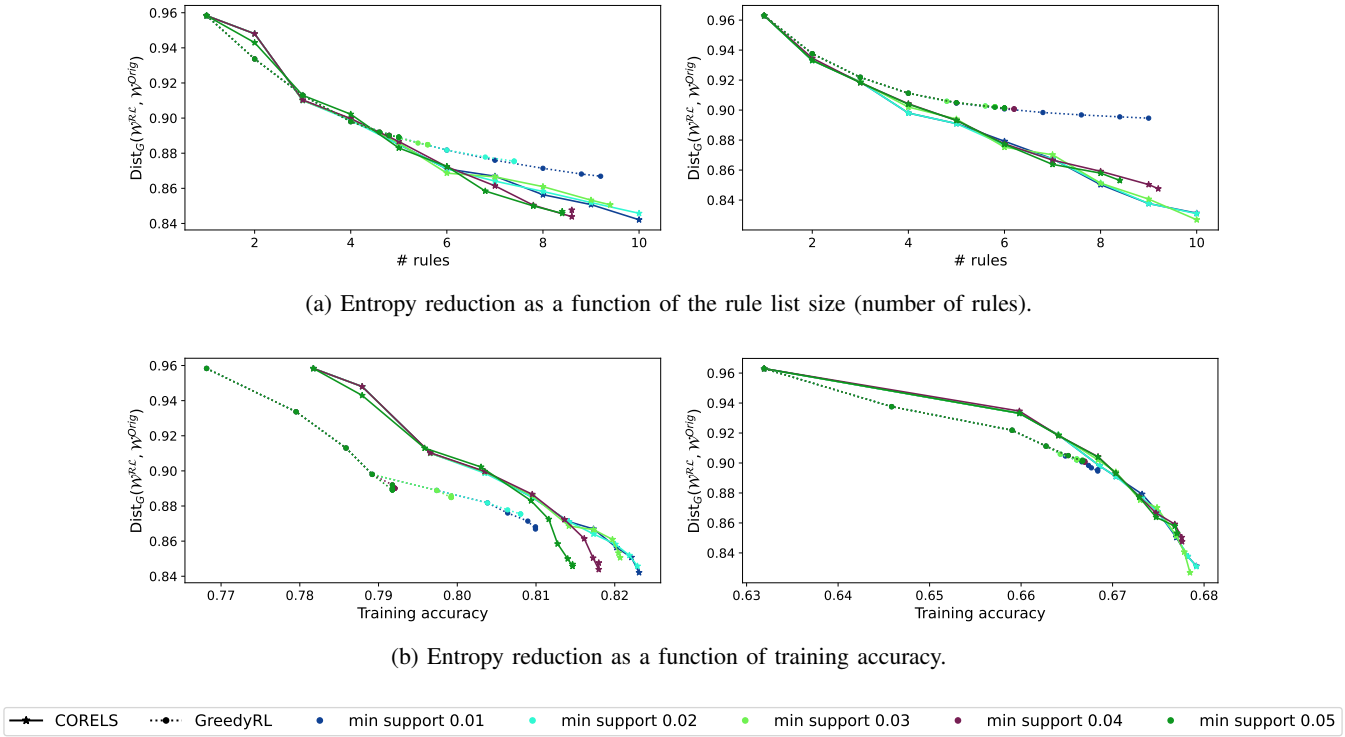
Fig. 2: Results of our experiments comparing optimal and greedily-built decision trees (learnt respectively with `DL8.5` and `sklearn_DT`), for different (relative) minimum leaf support values. Left: Adult Income dataset, Right: COMPAS dataset.

their training data. These observations can be explained by the nature of the learning algorithms. On the one side, greedy algorithms make heuristic choices iteratively. These choices are usually sub-optimal, and thus while leading to sub-optimal models (in terms of accuracy), they can also cause unnecessary leaks regarding their training data. On the other side, optimal learning algorithms perform global optimization and encode exactly the information needed in the most effective way.

For both datasets and types of models, the entropy reduction is not uniformly distributed across all training examples. Indeed, we plot in Figure 4 the minimum entropy reduction ratio as a function of the proportion of concerned training examples. One can observe that the amount of information contained by the learnt models varies significantly between different training examples. For instance, in the experiments using optimal rule lists with maximum size 10 and minimum support 0.01 on the Adult Income dataset (Figure 4b (left)), the less exposed training examples have an entropy reduction ratio above 0.95: knowledge of the rule list removes only 5% of the uncertainty regarding such samples. For the most exposed examples, this number becomes smaller than 0.60: knowledge of the rule list removes more then 40% of the uncertainty regarding such samples. This *disparate information leak* is intuitive: an example classified by a very long branch of a tree goes through many nodes, which gives more information regarding its features. This phenomenon is observed in all our experiments, with roughly identical distribution of the uncertainty reduction over the training datasets. It suggests that, behind average-case uncertainty reduction as reported in Figures 2 and 3, investigating per-example uncertainty reductions can also be insightful.

One can note that averaging the curves of Figure 4 leads to the computation of dataset-wide metrics as shown in Figures 2 and 3. For instance, we observe in Figure 4b that for most proportions of training samples, rule lists learnt using `CORELS` exhibit a lower entropy reduction ratio than those produced by `GreedyRL`. As aforementioned, these experiments use the largest considered rule lists (learned with maximum depth 10 and minimum support 0.01) for both methods, corresponding to the rightmost points on Figure 3a. Observing these particular points, one can see that rule lists built with `CORELS` indeed exhibit lower entropy reduction ratios than those built by `GreedyRL`, which is consistent with Figure 4b.

We observe a different trend for the decision trees learnt on the Adult Income dataset (Figure 4a (left)): the models built with the greedy `sklearn_DT` algorithm exhibit lower entropy reduction ratios than the optimal ones produced by `DL8.5`, hence containing more information. Again, these models correspond to the rightmost points on Figure 2a (left). For these experiments, the optimal models learnt with the `DL8.5` algorithm with the largest size constraint are indeed more compact than those produced by `sklearn_DT`



(a) Entropy reduction as a function of the rule list size (number of rules).

(b) Entropy reduction as a function of training accuracy.

Fig. 3: Results of our experiments comparing optimal and greedily-built rule lists (learnt respectively with `CORELS` and `GreedyRL`), for different (relative) minimum rule support values. Left: Adult Income dataset, Right: COMPAS dataset.

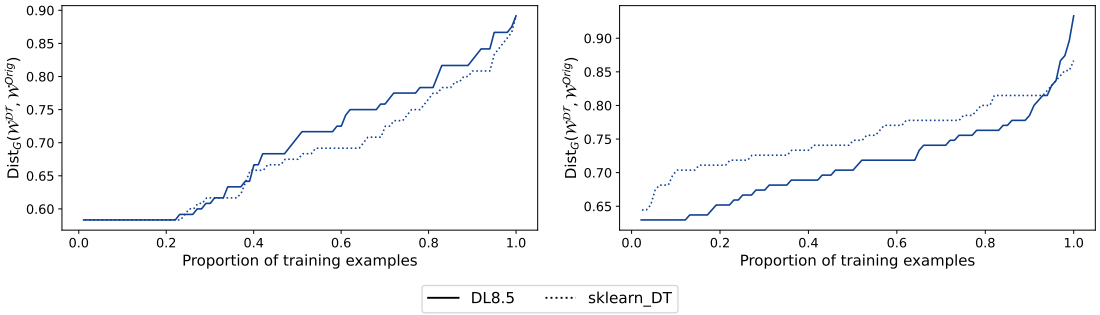
and contain less information overall. As aforementioned, this illustrates a drawback of greedy learning algorithms: by performing local (possibly sub-optimal) choices, they can produce models performing non-necessary or redundant operations, leaking additional information regarding their training data. This dimension is further explored in the Appendix A, where we relate the actual models' sizes and entropy reduction ratios to the constraints enforced during learning.

Finally, comparing decision trees and rule lists empirically as was done theoretically in Section IV-C could also be insightful. In particular, one could assess whether the rules' ordering, which allows the rules within a rule list to overlap (while the branches of a decision tree are all disjoint), empirically provides more information regarding the training data as was expected theoretically. However, such an experiment requires learning optimal rule lists whose rules' widths (*i.e.*, number of attributes involved in a rule's conjunction) match the depth of the tree's branches, which is computationally challenging. Indeed, considering sub-optimal models would bias the comparison as the results would depend on the performances of the learning algorithms rather than those of the models themselves.

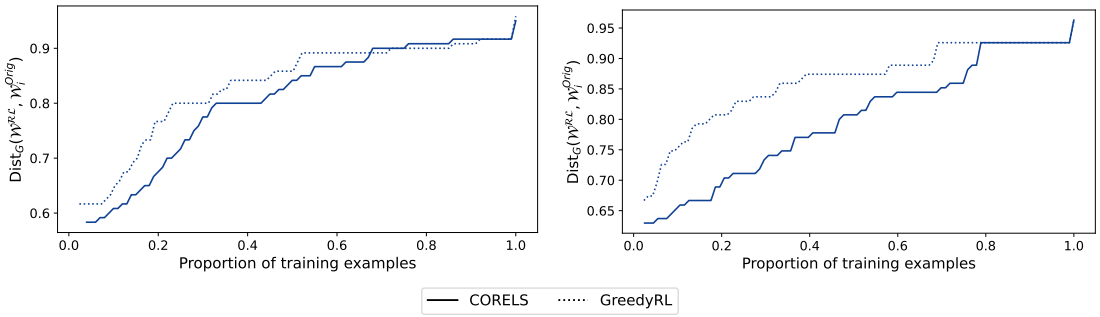
VI. Conclusion

We extended previous work and proposed generic tools to represent and precisely quantify the amount of information an interpretable model encodes regarding its training data. Such tools, and in particular the proposed generalized probabilistic datasets and the metric quantifying their amount of uncertainty, are generic enough to encode any type of knowledge - and hence are suitable to model a reconstructed dataset from any type of interpretable models. While their practical use may be computationally challenging in the general case, we demonstrated theoretically that they can be employed efficiently under reasonable assumptions. Furthermore, we empirically illustrated their usefulness through an example use case: assessing the effect of optimality in training machine learning models. Designing appropriate data structures to efficiently represent our generalized probabilistic datasets in the general case and handle more complex types of models such as random forests or other ensemble classifiers is an important challenge. However, this is out of the scope of this paper - as we restrict our attention to interpretable classifiers. Furthermore, generalized probabilistic datasets correspond to probabilistic databases, and so advances in this field will also enhance research in this direction. In the Appendix B, we further discuss how Dist_G can be interpreted and how it relates to other information theory metrics.

Interestingly, our approach can be seen as upper-bounding the capacity of any adversary (with the exact same information) for reconstructing the training dataset. Indeed, each (deterministic) reconstruction encoded via our (generalized) probabilistic dataset is compatible with the trained model's structure, hence without further information no adversary should be able to distinguish between them. The proposed



(a) Optimal and greedily-built decision trees, learned respectively with the `DL8.5` and `sklearn_DT` algorithms.



(b) Optimal and greedily-built rule lists, learned respectively with the `CORELS` and `GreedyRL` algorithms.

Fig. 4: Illustration of the *disparate information leak* phenomenon, for both optimal and greedily-built decision trees and rule lists, learned with the largest considered size constraints, *i.e.,* maximum depth 10 and minimum (relative) support 0.01. More precisely, we report the proportion of training examples for which the entropy reduction ratio is at most at a given value. Left: Adult Income dataset, Right: COMPAS dataset.

approach could also be leveraged to mount further inference attacks targeting the model's training data [18]. For instance, one could search within the encoded set of possible reconstructions whether a particular profile appears. If it never appears, then the result of the associated membership inference attack is certifiably negative. If it does, the proportion of possible reconstructions in which it exists can be used as an estimate of the probability of inclusion in the training data. Property inference attacks could also be conducted by averaging statistics over the set of possible reconstructions.

A promising extension of our study consists in leveraging the knowledge of the learning algorithm's internals to lower the reconstructed generalized probabilistic dataset entropy. For instance, if a greedy algorithm uses the Gini impurity as a splitting criterion, we know that at a given node no feature other than the chosen one can yield a better Gini impurity value in the training set. Additionally, optimality itself gives information: some combinations of the attributes not used within an optimal decision tree can be discarded if they could allow the building of a better decision tree. General demographic information could also help identify more probable reconstructions. Technically speaking, external knowledge could either discard some of the possible reconstructions or modify the probability distribution over them. Importantly, we do not consider it in this paper in order to remain algorithm-agnostic and focus on the interpretable models themselves.

We empirically observed that the entropy reduction brought by the knowledge of some interpretable model is not uniform across all examples. Investigating whether it disproportionately affects some subgroup of the population is an interesting direction. Another promising future work consists in combining the knowledge of different generalized probabilistic datasets, as was proposed in [1]. This would require aligning them, as well as merging several probability distributions, while in the original setup it simply consisted in union of sets. Finally, learning interpretable models optimizing our information leak measure could also constitute an interesting defense mechanism, in particular to avoid worst-case leakages as identified in our experiments. Investigating the effect of privacy-preserving methods such as the widely used *Differential Privacy* [46] on the quality of the probabilistic reconstruction is also an insightful research avenue. In the Appendix C, we overview challenges related to attacking differentially private models and discuss possible techniques to address them.

References

[1] S. Gambs, A. Gmati, and M. Hurfin, "Reconstruction attack through classifier analysis," in *Data and Applications Security and Privacy XXVI - 26th Annual IFIP WG 11.3 Conference, DBSec 2012, Paris, France, July 11-13,2012. Proceedings*, ser. Lecture Notes in Computer Science, N. Cuppens-Boulahia, F. Cuppens, and J. García-Alfaro, Eds., vol. 7371. Springer, 2012, pp. 274–281. [Online]. Available: https://doi.org/10.1007/978-3-642-31540-4_21

[2] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[3] M. Craven and J. Shavlik, "Extracting tree-structured representations of trained networks," *Advances in neural information processing systems*, vol. 8, 1995.

[4] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & explorable approximations of black box models," *arXiv preprint arXiv:1707.01154*, 2017.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[6] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[7] U. Aïvodji, H. Arai, O. Fortineau, S. Gambs, S. Hara, and A. Tapp, "Fairwashing: the risk of rationalization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 161–170.

[8] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 180–186.

[9] B. Dimanov, U. Bhatt, M. Jamnik, and A. Weller, "You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods." 2020.

[10] G. Laberge, U. Aïvodji, and S. Hara, "Fooling shap with stealthily biased sampling," *arXiv preprint arXiv:2205.15419*, 2022.

[11] U. Aïvodji, H. Arai, S. Gambs, and S. Hara, "Characterizing the risk of fairwashing," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 822–14 834, 2021.

[12] A. A. Freitas, "Comprehensible classification models: a position paper," *ACM SIGKDD explorations newsletter*, vol. 15, no. 1, pp. 1–10, 2014.

[13] R. L. Rivest, "Learning decision lists," *Mach. Learn.*, vol. 2, no. 3, pp. 229–246, 1987. [Online]. Available: https://doi.org/10.1007/BF00058680

[14] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin, "Learning certifiably optimal rule lists," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 35–44.

[15] P. R. Rijnbeek and J. A. Kors, "Finding a short and accurate decision rule in disjunctive normal form by exhaustive search," *Machine learning*, vol. 80, no. 1, pp. 33–62, 2010.

[16] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[17] C. Dwork, A. Smith, T. Steinke, and J. Ullman, "Exposed! a survey of attacks on private data," *Annual Review of Statistics and Its Application*, vol. 4, no. 1, pp. 61–84, 2017. [Online]. Available: https://doi.org/10.1146/annurev-statistics-060116-054123

[18] M. Rigaki and S. Garcia, "A survey of privacy attacks in machine learning," *CoRR*, vol. abs/2007.07646, 2020. [Online]. Available: https://arxiv.org/abs/2007.07646

[19] E. D. Cristofaro, "An overview of privacy in machine learning," *CoRR*, vol. abs/2005.08679, 2020. [Online]. Available: https://arxiv.org/abs/2005.08679

[20] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, F. Neven, C. Beeri, and T. Milo, Eds. ACM, 2003, pp. 202–210. [Online]. Available: https://doi.org/10.1145/773153.773173

[21] C. Dwork, F. McSherry, and K. Talwar, "The price of privacy and the limits of lp decoding," in *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 85–94. [Online]. Available: https://doi.org/10.1145/1250790.1250804

[22] A. Cohen and K. Nissim, "Linear program reconstruction in practice," *J. Priv. Confidentiality*, vol. 10, no. 1, 2020. [Online]. Available: https://doi.org/10.29012/jpc.711

[23] A. Gadotti, F. Houssiau, L. Rocher, B. Livshits, and Y. de Montjoye, "When the signal is in the noise: Exploiting diffix's sticky noise," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, 2019, pp. 1081–1098. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/gadotti

[24] S. Garfinkel, J. M. Abowd, and C. Martindale, "Understanding database reconstruction attacks on public data: These attacks on statistical databases are no longer a theoretical danger." *Queue*, vol. 16, no. 5, p. 28–53, oct 2018. [Online]. Available: https://doi.org/10.1145/3291276.3295691

[25] M. Fredrikson, E. Lantz, S. Jha, S. M. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, K. Fu and J. Jung, Eds. USENIX Association, 2014, pp. 17–32. [Online]. Available: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/fredrikson_matthew

[26] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, I. Ray, N. Li, and C. Kruegel, Eds. ACM, 2015, pp. 1322–1333. [Online]. Available: https://doi.org/10.1145/2810103.2813677

[27] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, Eds. ACM, 2017, pp. 587–601. [Online]. Available: https://doi.org/10.1145/3133956.3134077

[28] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, 2019, pp. 267–284. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/carlini

[29] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning: Revisited and enhanced," in *Applications and Techniques in Information Security - 8th International Conference, ATIS 2017, Auckland, New Zealand, July 6-7, 2017, Proceedings*, ser. Communications in Computer and Information Science, L. Batten, D. S. Kim, X. Zhang, and G. Li, Eds., vol. 719. Springer, 2017, pp. 100–110. [Online]. Available: https://doi.org/10.1007/978-981-10-5421-1_9

[30] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-leak: Data set inference and reconstruction attacks in online learning," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, S. Capkun and F. Roesner, Eds. USENIX Association, 2020, pp. 1291–1308. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/salem

[31] H. Hu and C. Lan, "Inference attack and defense on the distributed private fair learning framework," in *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, 2020.

[32] J. Aalmoes, V. Duddu, and A. Boutet, "Dikaios: Privacy auditing of algorithmic fairness via attribute inference attacks," *arXiv preprint arXiv:2202.02242*, 2022.

[33] F. Hamman, J. Chen, and S. Dutta, "Can querying for bias leak protected attributes? achieving privacy with smooth sensitivity," in *NeurIPS 2022 Workshop on Algorithmic Fairness through the Lens of Causality and Privacy*, 2022.

[34] J. Ferry, U. Aïvodji, S. Gambs, M.-J. Huguet, and M. Siala, "Exploiting fairness to enhance sensitive attributes reconstruction," in *First IEEE Conference on Secure and Trustworthy Machine Learning*, 2023. [Online]. Available: https://openreview.net/forum?id=tOVr0HLaFz0

[35] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California Law Review*, vol. 104, no. 3, pp. 671–732, 2016. [Online]. Available: http://www.jstor.org/stable/24758720

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duch-

esnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
[37] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
[38] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.
[39] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin, "Learning certifiably optimal rule lists for categorical data," *Journal of Machine Learning Research*, vol. 18, no. 234, pp. 1–78, 2018.
[40] D. Suciu, D. Olteanu, C. Ré, and C. Koch, *Probabilistic Databases*, ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. [Online]. Available: https://doi.org/10.2200/S00362ED1V01Y201105DTM016
[41] G. Aglin, S. Nijssen, and P. Schaus, "Learning optimal decision trees using caching branch-and-bound search," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 2020, pp. 3146–3153. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5711
[42] ——, "Pydl8.5: a library for learning optimal decision trees," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, C. Bessiere, Ed. ijcai.org, 2020, pp. 5222–5224. [Online]. Available: https://doi.org/10.24963/ijcai.2020/750
[43] C. Singh, K. Nasseri, Y. S. Tan, T. Tang, and B. Yu, "imodels: a python package for fitting interpretable models," p. 3192, 2021. [Online]. Available: https://doi.org/10.21105/joss.03192
[44] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[45] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. propublica (2016)," *ProPublica, May*, vol. 23, 2016.
[46] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
[47] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 493–502. [Online]. Available: https://doi.org/10.1145/1835804.1835868
[48] S. Fletcher and M. Z. Islam, "Decision tree classification with differential privacy: A survey," *ACM Comput. Surv.*, vol. 52, no. 4, pp. 83:1–83:33, 2019. [Online]. Available: https://doi.org/10.1145/3337064

Appendix A
Additional Results

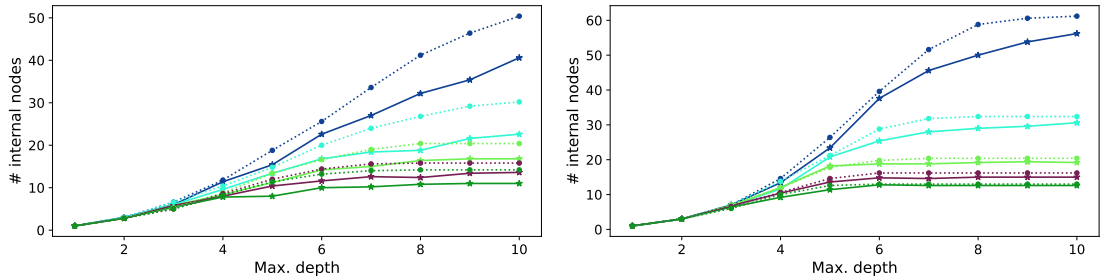
We observed in Section V-B that optimal models (either decision trees or rule lists) usually contain more information than greedily-built ones of the same size. However, when related to the models' utility (accuracy on the training data), this trend is reversed, and optimal models leak less information regarding their training data than greedily-built ones for the same performances level. This was explained by the fact that greedy learning algorithms iteratively make local choices that are sub-optimal, overall adding unnecessary information to the resulting model (*e.g.,* performing more splits than necessary within decision trees).

In this appendix section, we relate the amount of information an interpretable model carries to the size constraints that were enforced to build it. More precisely, we report in Figures 5a and 6a the resulting model size as a function of the maximum depth constraint, for the different minimum support constraints. Again, the model size is quantified as the number of internal nodes for a decision tree, or as the number of rules for width-1 rule lists. We also report in Figures 5b and 6b the overall entropy reduction ratio, as a function of the maximum depth constraint.

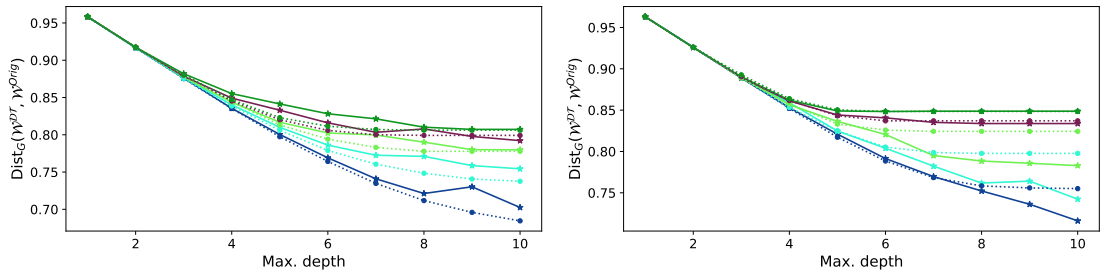
One can observe in Figure 5a that, as expected, the number of internal nodes within the built decision trees grows with the maximum depth value. Enforcing large values of the (relative) minimum leaf support quickly prevents the trees from expanding, as no split can be performed while satisfying the minimum support constraint. Hence, as expected, lowering the minimum support value leads to the computation of larger decision trees. Comparing greedily-built and optimal decision trees, one can note that the models learnt by `sklearn_DT` contain more nodes than the optimal ones built using `DL8.5`, for the same provided parameters (*i.e.,* minimum leaf support and maximum depth values). This can be explained by the fact that `sklearn_DT` often adds non-necessary splits as it iteratively performs local, sub-optimal choices. Meanwhile, many branches do not reach the enforced maximum depth within the optimal decision trees thanks to the performed global optimization which considers a split only if it is necessary. As a consequence, we observe in Figure 5b (left) that, for fixed parameters, the decision trees produced by `sklearn_DT` on the Adult Income dataset contain more information than those learnt by `DL8.5`. For the COMPAS dataset (Figure 5b (right)), we observe the opposite trend. This can be explained by two observations. First, the size difference between optimal and greedily-built decision trees is smaller on the COMPAS dataset (Figure 5a (right)) than on the Adult dataset (Figure 5a (left)). Then, in average, we saw within Section V-B (Figure 2a) that, for equivalent sizes, the optimal decision trees carry more information than the greedily-built ones.

Figure 6a shows that, as expected, the number of rules within the built rule lists grows with the enforced maximum depth value. As for the decision trees, largest values of the enforced minimum rule support prevent expansion of the rule lists, when no rule satisfying the minimum support constraint can be found. This is particularly true for the greedy learning algorithm. Indeed, at each iteration, the algorithm selects a rule maximizing a given criterion (*i.e.,* minimizing Gini Impurity). Then, the examples not captured by the rules fall into the rest of the rule list, and are used for the next iterations. If the algorithm selects rules with large supports during the first iterations, there may be too few remaining examples to be able to add new rules. This drawback is not observed with `CORELS`, as it performs global optimization. As a direct consequence, one can see in Figure 6b that, for fixed parameters (*i.e.,* minimum rule support and maximum depth values), the rule lists built using `CORELS` contain more information than those produced by `GreedyRL`. This trend is related to the observed size difference, but is also exacerbated by the fact that, as observed in Section V-B (Figure 3a), optimal rule lists usually encode more information that greedily-built ones of equivalent size.

Finally, we report in Figure 7 the entropy reduction ratio as a function of the minimum (leaf or rule) support constraint, for optimal and greedily-built decision trees (Figure 7a) or rule



(a) Experiments relating the actual models' sizes to the size constraints enforced during learning. We report tree size (number of splits/internal nodes) as a function of the maximum depth constraint.



(b) Experiments relating the entropy reduction ratio to the size constraints enforced during learning. We report the entropy reduction as a function of the maximum depth constraint.

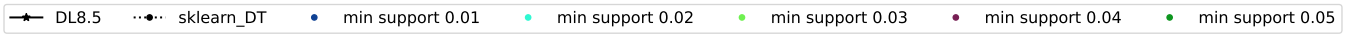


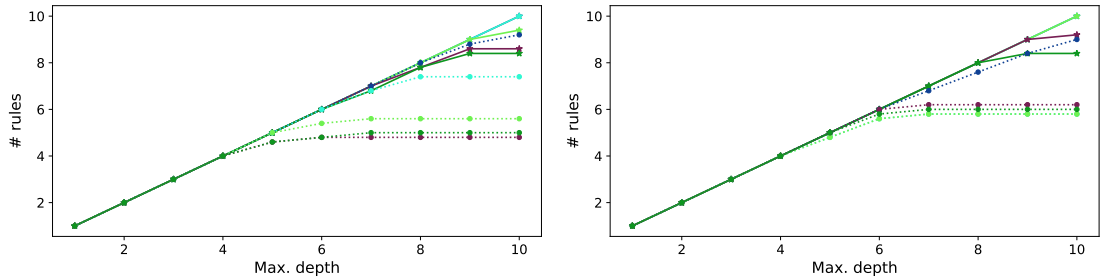
Fig. 5: Results of our experiments comparing optimal and greedily-built decision trees (learnt respectively with `DL8.5` and `sklearn_DT`), for different (relative) minimum leaf support values. Left: Adult Income dataset, Right: COMPAS dataset.

lists (Figure 7b). For decision tree models, one can observe that the minimum leaf support value only influences learning for large enough maximum depth constraints. In such case, smaller minimum support values allow the building of longer branches, which results in decision trees containing more information. The trend is not that clear for rule lists models, which is explained by two observations. First, contrary to decision trees, rule lists exactly matching the maximum depth constraint can almost always be built using `CORELS`, which performs global optimization, as can be observed in Figure 6a. In such a case, the minimum support value does not actually influence the size of the learnt model. Second, while the leaves of a decision tree have disjoint supports, this is not the case for the successive rules within a rule list, and enforcing greater support for the first rules impacts the possible support values for the following ones. Overall, the relationship between the minimum rule support constraint and the entropy reduction ratio is complex, and no clear trend can be observed. On the opposite, the minimum leaf support constraint used when building a decision tree strongly influences the tree size and the amount of information it contains.

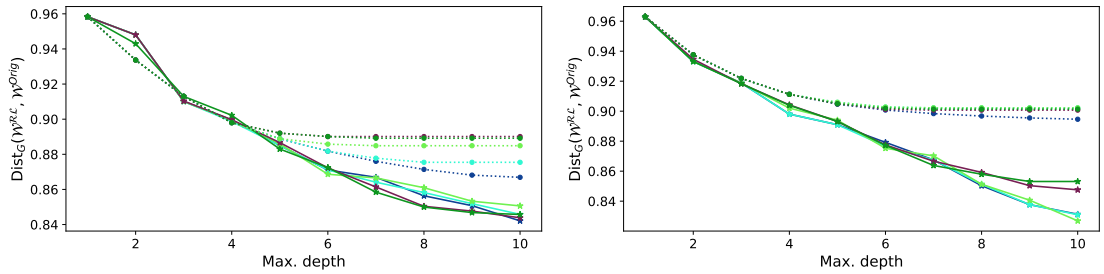
Appendix B
Discussion on the Proposed Attack Success Metric

The objective of the considered adversary is to reconstruct the entire training dataset $\mathcal{W}^{Orig}$, *i.e.,* all attributes of all examples: $w^{Orig}_{i\in[1..n],k\in[1..d]}$. Thus, from the perspective of an uninformed adversary (who only knows the training set size n and the domains $\mathcal{V}_k$ of the different attributes $k \in [1..d]$) all the possible combinations of all attributes' values (within their domains) are possible for every example.

Any knowledge reducing the scope of possible reconstructions hence eases the adversary's task and raises the probability of correct reconstruction if the adversary chooses among the (reduced) set of possible reconstructions. This reduction is precisely what our proposed metric, $\mathsf{Dist}_G(\mathcal{W}^M, \mathcal{W}^{Orig})$, quantifies, through the ratio of the (joint) entropy of the (generalized) probabilistic dataset over the (joint) entropy of the uninformed reconstruction. Intuitively, it measures the uncertainty remaining in the built (generalized) probabilistic reconstruction $\mathcal{W}^M$, through the joint entropy of all the random variables $\mathcal{W}^M_{i\in[1..n],k\in[1..d]}$ modeling the dataset. As all reconstructions are compatible with the provided model's structure, and the true (unknown) training dataset is provably part of the encoded set of possible reconstructions, this uncertainty measure is a good indicator on how much we can reduce



(a) Experiments relating the actual models' sizes to the size constraints enforced during learning. We report rule list size (number of rules) as a function of the maximum depth constraint.



(b) Experiments relating the entropy reduction ratio to the size constraints enforced during learning. We report the entropy reduction as a function of the maximum depth constraint.

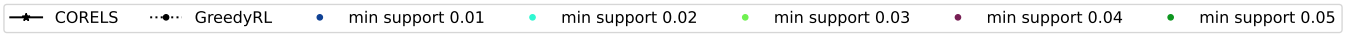


Fig. 6: Results of our experiments comparing optimal and greedily-built rule lists (learnt respectively with `CORELS` and `GreedyRL`), for different (relative) minimum rule support values. Left: Adult Income dataset, Right: COMPAS dataset.

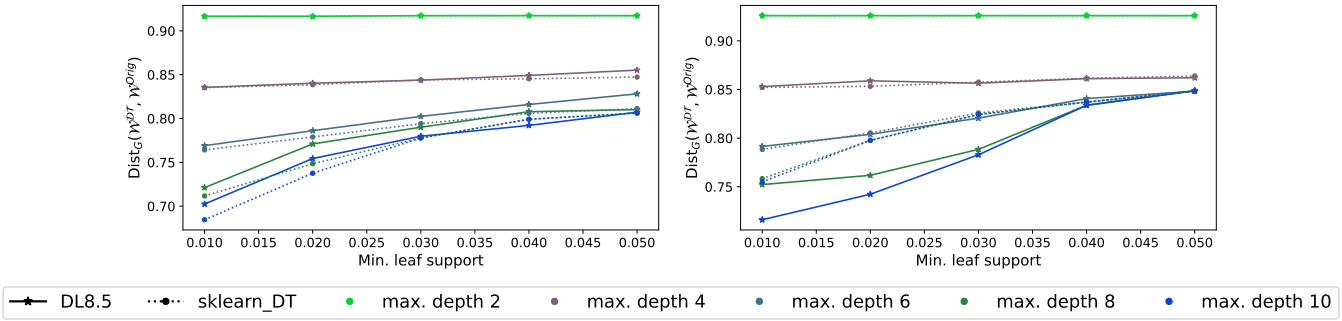
the space of possible solutions around the correct one. In other words, Dist_G can be seen as a measure of reconstructibility for the whole dataset.

One could note that the mutual information between the input database and the (generalized) probabilistic reconstruction may be an interesting indicator. However, its computation would reduce to the probability of the true input database in the built (generalized) probabilistic reconstruction $\mathbb{P}\left(\mathcal{W}_{i\in[1..n],k\in[1..d]} = w^{Orig}_{i\in[1..n],k\in[1..d]}\right)$. While this indicator would also be insightful, it would ignore the other reconstructions that are compatible with the model's structure. In practice, interpretable models (in particular, decision trees and rule lists) allow the building of a set of possible reconstructions, but without further information, one cannot distinguish between them. In a sense, they are all equally probable and the probability of the correct reconstruction is exactly $\frac{1}{|\Pi(\mathcal{W}^M)|}$, which goes back to our proposed computation (Section IV). Formulating other reconstructibility criteria leveraging tools from the information theory is an interesting research avenue, building upon our thorough formalization of the notion of (generalized) probabilistic datasets.

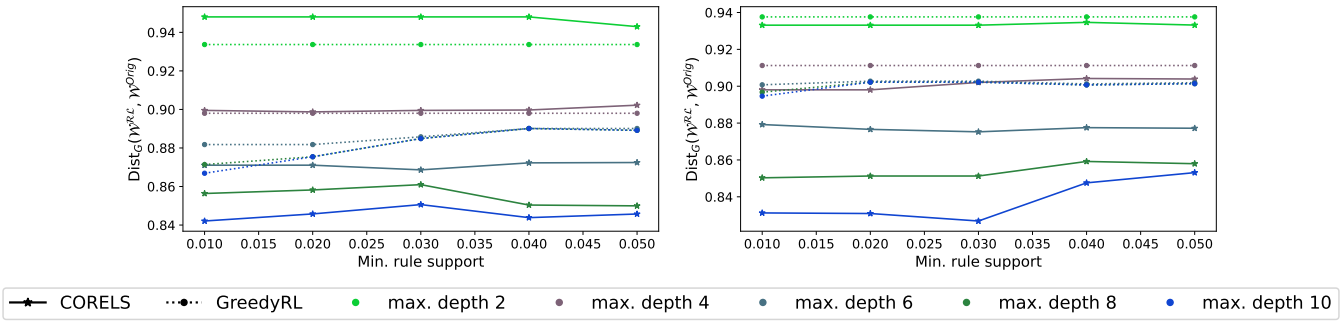
Appendix C
(Generalized) Probabilistic Reconstruction from Differentially Private Models: Challenges

Assessing the effect of Differential Privacy (DP) [46] on the proposed probabilistic reconstruction would be insightful. This would require building interpretable models using differentially private learning algorithms. In particular, varying the privacy budget and observing the (potential) impact on the resulting probabilistic reconstruction success would be interesting to assess whether (and for which parameters) DP provides a valuable protection against such dataset-level attacks.

For instance, one could consider differentially private decision trees [47], [48]. Note that the rationale of the following discussion also holds for rule lists or other types of interpretable models. Recall that the aim of our approach is to construct an object (a (generalized) probabilistic dataset) encoding the whole set of dataset reconstructions that are compatible with the structure of a given (trained) model. One key challenge is that DP decision trees can only provide noisy counts within the tree leaves. Indeed, such counts are required to compute confidence scores, but the exact per-class cardinalities (as displayed within the decision tree represented in Figure 1) would violate DP. These noisy counts, if used directly in the proposed iterative reconstruction process (which



(a) Optimal and greedily-built decision trees, learned respectively with the `DL8.5` and `sklearn_DT` algorithms.



(b) Optimal and greedily-built rule lists, learned respectively with the `CORELS` and `GreedyRL` algorithms.

Fig. 7: Results of our experiments relating the entropy reduction ratio to the (relative) minimum support constraints enforced during learning. We report the entropy reduction as a function of the (relative) minimum (leaf or rule) support constraint, for different maximum depth constraint values. Left: Adult Income dataset, Right: COMPAS dataset.

follows the path through each branch to perform the attributes' domains reductions as specified by the internal nodes' splits), would then induce errors in the performed reconstruction. In the worst case, the true reconstruction could not be part of the possible ones (represented using the constructed (generalized) probabilistic dataset) due to these errors. In such a case, one can still use the (generalized) probabilistic dataset variables' joint entropy to estimate the uncertainty within the encoded set of possible reconstructions, but measuring the distance between the true dataset and the set of plausible reconstructions given the DP model would also be an interesting indicator.

Considering other types of models (such as rule sets), DP could even lead to infeasible reconstruction problems (*i.e.,* empty set of possible reconstructions) due to inconsistent (noisy) counts within the different rules. A promising mitigation technique would be to account for the added noise, for instance by modeling the noise values using linear programming as was done in previous works [22]. However, this requires the use of more elaborated techniques and is out of the scope of this paper.