The VLDB Journal (2008) 17:789–804
DOI 10.1007/s00778-006-0039-5

REGULAR PAPER

Providing *k*-anonymity in data mining

Arik Friedman · Ran Wolff · Assaf Schuster

Received: 30 September 2005 / Revised: 24 May 2006 / Accepted: 2 August 2006 / Published online: 10 January 2007

Abstract In this paper we present extended definitions of *k*-anonymity and use them to prove that a given data mining model does not violate the *k*-anonymity of the individuals represented in the learning examples. Our extension provides a tool that measures the amount of anonymity retained during data mining. We show that our model can be applied to various data mining problems, such as classification, association rule mining and clustering. We describe two data mining algorithms which exploit our extension to guarantee they will generate only *k*-anonymous output, and provide experimental results for one of them. Finally, we show that our method contributes new and efficient ways to anonymize data and preserve patterns during anonymization.

1 Introduction

In recent years the data mining community has faced a new challenge. Having shown how effective its tools are in revealing the knowledge locked within huge databases, it is now required to develop methods that restrain the power of these tools to protect the privacy of individuals. This requirement arises from popular concern about the powers of large corporations and government agencies—concern which has been reflected in the actions of legislative bodies (e.g., the debate about and subsequent elimination of the Total Information Awareness project in the US [10]). In an odd turn of events, the same corporations and government organizations which are the cause of concern are also among the main pursuers of such privacy-preserving methodologies. This is because of their pressing need to cooperate with each other on many data analytic tasks (e.g., for cooperative cyber-security systems, failure analysis in integrative products, detection of multilateral fraud schemes, and the like).

The first approach toward privacy protection in data mining was to perturb the input (the data) before it is mined [4]. Thus, it was claimed, the original data would remain secret, while the added noise would average out in the output. This approach has the benefit of simplicity. At the same time, it takes advantage of the statistical nature of data mining and directly protects the privacy of the data. The drawback of the perturbation approach is that it lacks a formal framework for proving how much privacy is guaranteed. This lack has been exacerbated by some recent evidence that for some data, and some kinds of noise, perturbation provides no privacy at all [20, 24]. Recent models for studying the privacy attainable through perturbation [9,11,12,15,17] offer solutions to this problem in the context of statistical databases.

At the same time, a second branch of privacy preserving data mining was developed, using cryptographic techniques. This branch became hugely popular [14,19, 22,27,37,42] for two main reasons: First, cryptography offers a well-defined model for privacy, which includes methodologies for proving and quantifying it. Second, there exists a vast toolset of cryptographic algorithms and constructs for implementing privacy-preserving data mining algorithms. However, recent work (e.g., [14,23])

A. Friedman (✉) · A. Schuster
Computer Science Department,
Technion—Israel Institute of Technology, Haifa, Israel
e-mail: arikf@cs.technion.ac.il

A. Schuster
e-mail: assaf@cs.technion.ac.il

R. Wolff
Management Information Systems Department,
Haifa University, Haifa, Israel
e-mail: rwolff@mis.haifa.ac.il

has pointed that cryptography does not protect the output of a computation. Instead, it prevents privacy leaks in the *process* of computation. Thus, it falls short of providing a complete answer to the problem of privacy preserving data mining.

One definition of privacy which has come a long way in the public arena and is accepted today by both legislators and corporations is that of k-anonymity [34]. The guarantee given by k-anonymity is that no information can be linked to groups of less than k individuals. The k-anonymity model of privacy was studied intensively in the context of public data releases [3,7,8,18,21,26,29,32,33], when the database owner wishes to ensure that no one will be able to link information gleaned from the database to individuals from whom the data has been collected. In the next section we provide, for completeness, the basic concepts of this approach.

We focus on the problem of guaranteeing privacy of data mining output. To be of any practical value, the definition of privacy must satisfy the needs of users of a reasonable application. Two examples of such applications are (1) a credit giver, whose clientele consists of numerous shops and small businesses, and who wants to provide them with a classifier that will distinguish creditworthy from credit-risky clients, and (2) a medical company that wishes to publish a study identifying clusters of patients who respond differently to a course of treatment. These data owners wish to release data mining output, but still be assured that they are not giving away the identity of their clients. If it could be verified that the released output withstands limitations similar to those set by k-anonymity, then the credit giver could release a k-anonymous classifier and reliably claim that the privacy of individuals is protected. Likewise, the authors of a medical study quoting k-anonymous cluster centroids could be sure that they comply with HIPAA privacy standards [36], which forbid the release of individually identifiable health information.

One way to guarantee k-anonymity of a data mining model is to build it from a k-anonymized table. However, this poses two main problems: first, the performance cost of the anonymization process may be very high, especially for large and sparse databases. In fact, the cost of anonymization can exceed the cost of mining the data. Second, the process of anonymization may inadvertently delete features that are critical for the success of data mining and leave out those that are useless; thus, it would make more sense to perform data mining first and anonymization later.

To demonstrate the second problem, consider the data in Table 1, which describes loan risk information of a mortgage company. The *Gender*, *Married*, *Age* and *Sports Car* attributes contain data that is available to the public, while the *Loan Risk* attribute contains data that is known only to the company. To get a 2-anonymous version of this table, many practical methods call for the suppression or generalization of whole columns. This approach was termed *single-dimension recoding* [25]. In the case of Table 1, the data owner would have to choose between suppressing the *Gender* column and suppressing all the other columns.

Table 1 Mortgage company data

Name	Gender	Married	Age	Sports car	Loan risk
Anthony	Male	Yes	Young	Yes	Good
Brian	Male	Yes	Young	No	Good
Charles	Male	Yes	Young	Yes	Good
David	Male	Yes	Old	Yes	Good
Edward	Male	Yes	Old	Yes	Bad
Frank	Male	No	Old	Yes	Bad
Alice	Female	No	Young	No	Good
Barbara	Female	No	Old	Yes	Good
Carol	Female	No	Young	No	Bad
Donna	Female	Yes	Young	No	Bad
Emily	Female	Yes	Young	Yes	Bad
Fiona	Female	Yes	Young	Yes	Bad

Table 2 Anonymized mortgage company data

Gender	Married	Age	Sports car	Loan risk
Male	*	Young	*	Good
Male	*	Young	*	Good
Male	*	Young	*	Good
Male	*	Old	*	Good
Male	*	Old	*	Bad
Male	*	Old	*	Bad
Female	No	*	*	Good
Female	No	*	*	Good
Female	No	*	*	Bad
Female	Yes	*	*	Bad
Female	Yes	*	*	Bad
Female	Yes	*	*	Bad

The methods we describe in this paper would lead to full suppression of the *Sports Car* column as well as a partial suppression of the *Age* and *Married* columns. This would result in Table 2. This kind of generalization was termed *multi-dimensional recoding* [25]. While more data is suppressed, the accuracy of the decision tree learned from this table (Fig. 1) is better than that of the decision tree learned from the table without the *Gender* column. Specifically, without the *Gender* column, it is impossible to obtain a classification better than 50% *good* loan risk, 50% *bad* loan risk, for any set of tuples.

In this paper we extend the definition of k-anonymity with definitions of our own, which can then be used to prove that a given data mining model is k-anonymous. The key for these extended definitions is in

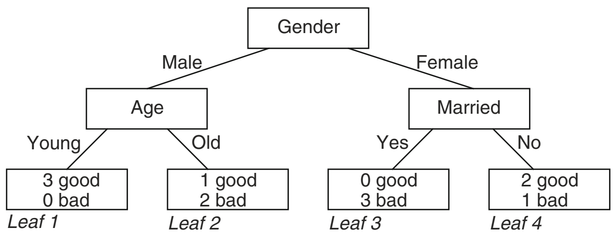

Fig. 1 Mortgage company decision tree

identifying how external data can be used to perform a linking attack on a released model. We exemplify how our definitions can be used to validate the *k*-anonymity of classification, clustering, and association rule models, and demonstrate how the definitions can be incorporated within a data mining algorithm to guarantee *k*-anonymous output. This method ensures the *k*-anonymity of the results while avoiding the problems detailed above.

This paper is organized as follows: In Sect. 2 we reiterate and discuss Sweeney's and Samarati's formal definition of *k*-anonymity. We then proceed in Sect. 3 to extend their definition with our definitions for *k*-anonymous data mining models. In Sect. 4 we exemplify the use of these definitions, and we present two *k*-anonymous data mining algorithms in Sect. 5. Section 6 shows experimental results using one of the algorithms from Sect. 5. Section 7 discusses related work. We present our conclusions in Sect. 8.

2 *k*-Anonymity of tables

The *k*-anonymity model was first described by Sweeney and Samarati [32], and later expanded by Sweeney [34, 31] in the context of data table releases. In this section we reiterate their definition and then proceed to analyze the merits and shortcomings of *k*-anonymity as a privacy model.

The *k*-anonymity model distinguishes three entities: individuals, whose privacy needs to be protected; the database owner, who controls a table in which each row (also referred to as record or tuple) describes exactly one individual; and the attacker. The *k*-anonymity model makes two major assumptions:

1. The database owner is able to separate the columns of the table into a set of *quasi-identifiers*, which are attributes that may appear in external tables the database owner does not control, and a set of private columns, the values of which need to be protected. We prefer to term these two sets as *public* attributes and *private* attributes, respectively.

Table 3 Table anonymization

(A) Original		(B) 2-Anonymized		(C) Public	
Zipcode	Income	Zipcode	Income	Zipcode	Name
11001	High	110XX	High	11001	John
11001	Low	110XX	Low	11001	Lisa
12033	Mid	120XX	Mid	12033	Ben
12045	High	120XX	High	12045	Laura

2. The attacker has full knowledge of the public attribute values of individuals, and no knowledge of their private data. The attacker only performs linking attacks. A linking attack is executed by taking external tables containing the identities of individuals, and some or all of the public attributes. When the public attributes of an individual match the public attributes that appear in a row of a table released by the database owner, then we say that the individual is linked to that row. Specifically the individual is linked to the private attribute values that appear in that row. A linking attack will succeed if the attacker is able to match the identity of an individual against the value of a private attribute.

As accepted in other privacy models (e.g., cryptography), it is assumed that the domain of the data (the attributes and the ranges of their values) and the algorithms used for anonymization are known to the attacker. Ignoring this assumption amounts to "security by obscurity," which would considerably weaken the model. The assumption reflects the fact that knowledge about the nature of the domain is usually public and in any case of a different nature than specific knowledge about individuals. For instance, knowing that every person has a height between zero and three meters is different than knowing the height of a given individual.

Under the *k*-anonymity model, the database owner retains the *k*-anonymity of individuals if none of them can be linked with fewer than *k* rows in a released table. This is achieved by making certain that in any table released by the owner there are at least *k* rows with the same combination of values in the public attributes. Since that would not necessarily hold for every table, most of the work under the *k*-anonymity model [7,18,21, 32–34] focuses on methods of suppressing, altering, and eliminating attribute values in order that the changed table qualify as *k*-anonymous.

Table 3 illustrates how *k*-anonymization hinders linking attacks. The joining of the original Table 3(A) with the public census data in 3(C) would reveal that *Laura*'s income is *High* and *Ben*'s is *Middle*. However, if the original table is 2-anonymized to that in 3(B), then the outcome of joining it with the census data is ambiguous.

It should be noted that the k-anonymity model is slightly broader than what is described here [34], especially with regard to subsequent releases of data. We chose to provide the minimal set of definitions required to extend k-anonymity in the next section.

2.1 The k-anonymity model: pros and cons

The limitations of the k-anonymity model stem from the two assumptions above. First, it may be very hard for the owner of a database to determine which of the attributes are or are not available in external tables. This limitation can be overcome by adopting a strict approach that assumes much of the data is public. The second limitation is much harsher. The k-anonymity model assumes a certain method of attack, while in real scenarios there is no reason why the attacker should not try other methods, such as injecting false rows (which refer to no real individuals) into the database. Of course, it can be claimed that other accepted models pose similar limitations. For instance, the well-accepted model of semi-honest attackers in cryptography also restricts the actions of the attacker.

A third limitation of the k-anonymity model published recently in the literature [28] is its implicit assumption that tuples with similar public attribute values will have different private attribute values. Even if the attacker knows the set of private attribute values that match a set of k individuals, the assumption remains that he does not know which value matches any individual in particular. However, it may well happen that, since there is no explicit restriction forbidding it, the value of a private attribute will be the same for an identifiable group of k individuals. In that case, the k-anonymity model would permit the attacker to discover the value of an individual's private attribute.

Despite these limitations, k-anonymity is one of the most accepted models for privacy in real-life applications, and provides the theoretical basis for privacy related legislation [36]. This is for several important reasons: (1) The k-anonymity model defines the privacy of the output of a process and not of the process itself. This is in sharp contrast to the vast majority of privacy models that were suggested earlier, and it is in this sense of privacy that clients are usually interested. (2) It is a simple, intuitive, and well-understood model. Thus, it appeals to the non-expert who is the end client of the model. (3) Although the process of computing a k-anonymous table may be quite hard [3,29], it is easy to validate that an outcome is indeed k-anonymous. Hence, non-expert data owners are easily assured that they are using the model properly. (4) The assumptions regarding separation of quasi-identifiers, mode of attack, and variability of private data have so far withstood the test of real-life scenarios.

3 Extending k-anonymity to models

We are now ready to present the first contribution of this paper: an extension of the definition of k-anonymity beyond the release of tables. Our definitions are accompanied by a simple example to facilitate comprehension.

Consider a mortgage company that uses a table of past borrowers' data to build a decision tree classifier predicting whether a client would default on the loan. Wishing to attract good clients and deter bad ones, the company includes the classifier on its Web page and allows potential clients to evaluate their chances of getting a mortgage. However, it would be unacceptable if somebody could use the decision tree to find out which past clients failed to return their loans. The company assumes that all the borrowers' attributes (age, marital status, ownership of a sports car, etc.) are available to an attacker, except for the *Loan Risk* attribute (*good/bad* loan risk), which is private.

Figure 1 describes a toy example of the company's decision tree, as induced from a set of learning examples, given in Table 1, pertaining to 12 past clients. We now describe a table which is equivalent to this decision tree in the sense that the table is built from the tree and the tree can be reconstructed from the table. The equivalent table (Table 2) has a column for each attribute that is used in the decision tree and a row for each learning example. Whenever the tree does not specify a value (e.g., *Marital Status* for male clients), the value assigned to the row will be *.

The motivation for the definitions which follow is that if the equivalent table is k-anonymous, the decision tree should be considered to be "k-anonymous" as well. The rationale is that, because the decision tree in Figure 1 can be reconstructed from Table 2, it contains no further information. Thus, if a linking attack on the table fails, any similar attack on the decision tree would have to fail as well. This idea that a data mining model and a k-anonymous table are equivalent allows us to define k-anonymity in the context of a broad range of models. We begin our discussion by defining a private database and then defining a model of that database.

Definition 1 (A private database) A private database T is a collection of tuples from a domain $D = A \times B = A_1 \times \cdots \times A_k \times B_1 \times \cdots \times B_\ell$. $A_1, \ldots, A_k$ are public attributes (a.k.a. quasi-identifiers) and $B_1, \ldots, B_\ell$ are private attributes.

We denote $A = A_1 \times \cdots \times A_k$ the *public subdomain* of D. For every tuple $x \in D$, the projection of x into A, denoted x_A, is the tuple in A that has the same assignment to each public attribute as x. The projection of a table T into A is denoted $T_A = \{x_A : x \in T\}$.

Definition 2 (A model) A model M is a function from a domain D to an arbitrary output domain O.

Every model induces an equivalence relation on D, i.e., $\forall x, y \in D, x \equiv y \Leftrightarrow M(x) = M(y)$. The model partitions D into respective equivalence classes such that $[x] = \{y \in D : y \equiv x\}$.

In the mortgage company decision tree example, the decision tree is a function that assigns bins to tuples in T. Accordingly, every bin within every leaf constitutes an equivalence class. Two tuples which fit into the same bin cannot be distinguished from one another using the tree, even if they do not agree on all attribute values. For example, although the tuples of *Anthony* and *Brian* do not share the same value for the *Sports Car* attribute, they both belong to the *good loan risk* bin of *leaf 1*. This is because the tree does not differentiate tuples according to the *Sports Car* attribute. On the other hand, while the tuples of *David* and *Edward* will both be routed to *leaf 2*, they belong to different bins because their loan risk classifications are different.

The model alone imposes some structure *on the domain*. However, when a data owner releases a model based on a database, it also provides information about how the model relates to the database. For instance, a decision tree model or a set of association rules may include the number of learning examples associated with each leaf, or the support of each rule, respectively. As we shall see, a linking attack can be carried out using the partitioning of the domain, together with the released populations of different regions.

Definition 3 (A release) Given a database T and a model M, a release M_T is the pair (M, p_T), where p_T (for population) is a function that assigns to each equivalence class induced by M the number of tuples from T that belong to it, i.e., $p_T([x]) = |T \bigcap [x]|$.

Note that other definitions of a release, in which the kind of information provided by p_T is different, are possible as well. For example, a decision tree may provide the relative frequency of a bin within a leaf, or just denote the bin that constitutes the majority class. In this paper we assume the worst case, in which the exact number of learning examples in each bin is provided. The effect of different kinds of release functions on the extent of private data that can be inferred by an attacker is an open question. Nevertheless, the anonymity analysis provided herein can be applied in the same manner for all of them. In other words, different definitions of p_T would reveal different private information on the same groups of tuples.

As described, the released model partitions the domain according to the values of public and private attributes. This is reasonable because the users of the model are intended to be the database owner or the client, both of whom supposedly know the private attributes' values. We now turn to see how the database and the release are perceived by an attacker.

Definition 4 (A public identifiable database) A public identifiable database $T_{\text{ID}} = \{(\text{id}_x, x_A) : x \in T\}$ is a projection of a private database T into the public subdomain A, such that every tuple of T_A is associated with the identity of the individual to whom the original tuple in T pertained.

Although the attacker knows only the values of public attributes, he can nevertheless try to use the release M_T to expose private information of individuals represented in T_{ID}. Given a tuple $(\text{id}_x, x_A) \in T_{\text{ID}}$ and a release, the attacker can distinguish the equivalence classes to which the original tuple x may belong. We call this set of equivalence classes the *span* of x_A.

Definition 5 (A span) Given a model M, the span of a tuple $a \in A$ is the set of equivalence classes induced by M, which contain tuples $x \in D$, whose projection into A is a. Formally, $S_M(a) = \{[x] : x \in D \wedge x_A = a\}$. When M is evident from the context, we will use the notation $S(a)$.

In the aforementioned mortgage company's decision tree model, every leaf constitutes a span, because tuples can be routed to different bins within a leaf by changing their private *Loan Risk* attribute, but cannot be routed to other leaves unless the value of a public attribute is changed. For example, an attacker can use the public attributes *Gender* and *Married* to conclude that the tuples of *Barbara* and *Carol* both belong to *leaf 4*. However, although these tuples have different values for the public attributes *Age* and *Sports Car*, the attacker cannot use this knowledge to determine which tuple belongs to which bin. These tuples are indistinguishable from the attacker's point of view, with respect to the model: both share the same span formed by *leaf 4*.

We will now consider the connection between the number of equivalence classes in a span and the private information that can be inferred from the span.

Claim 1 If $S(a)$ contains more than one equivalence class, then for every two equivalence classes in the span, $[x]$ and $[y]$, there is at least one combination of attribute values that appears in $[x]$ and does not appear in $[y]$.

Proof By definition, for every equivalence class $[x] \in S(a)$, there exists $x \in [x]$ such that $x_A = a$. Let $[x]$ and

$[y]$ be two equivalence classes in $S(a)$, and let $x \in [x]$, $y \in [y]$ be two tuples such that $x_A = y_A = a$. Since x and y have the same public attribute values, the only way to distinguish between them is by their private attribute values. The equivalence classes $[x]$ and $[y]$ are disjoint; hence, the combination of public and private attribute values x is not possible for $[y]$, and the combination of public and private attribute values y is not possible for $[x]$.

Given a pair $(\mathrm{id}_z, z_A) \in T_{\mathrm{ID}}$ such that $z_A = a$, the values of $p_T([x])$ and $p_T([y])$ allow the combination of private attributes possible for z to be exposed. For example, if $p_T([x]) = 0$, the attacker can rule out the possibility that $z = x$.

Claim 2 If $S(a)$ contains exactly one equivalence class, then no combination of private attributes can be eliminated for any tuple that has the same span.

Proof Let $x_A \in A$ be a tuple such that $S(x_A) = S(a)$. Let $y, z \in D$ be two tuples such that $y_A = z_A = x_A$. Regardless of the private attribute values of y and z, it holds that $[y] = [z]$. Otherwise, $S(x_A)$ would contain more than a single equivalence class, in contradiction to the assumption. Therefore, y and z both represent equally possible combinations of private attribute values for the tuple x_A, regardless of the population function p_T.

Corollary 1 A release exposes private information on the population of a span if and only if the span contains more than one equivalence class.

We will now see exactly how a release can be exploited to infer private knowledge about individuals. Given a public identifiable database T_{ID} and a model M, we use $S(a)_{T_{\mathrm{ID}}} = \{(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}} : S(x_A) = S(a)\}\}$ to denote the set of tuples that appear in T_{ID} and whose span is $S(a)$. These are tuples from T_{ID} which are indistinguishable with respect to the model M—each of them is associated with the same set of equivalence classes. Knowing the values of p_T for each equivalence class in $S(a)$ would allow an attacker to constrain the possible private attribute value combinations for the tuples in $S(a)_{T_{\mathrm{ID}}}$. For example, in the mortgage company's decision tree, the span represented by *leaf 4* (*Female, Unmarried*) contains two equivalence classes, which differ on the private attribute *Loan Risk*. Tuples that belong to the *good* equivalence class cannot have the private attribute *bad Loan Risk*, and vice versa. Given tuples that belong to a span with more than one equivalence class, the populations of each can be used to constrain the possible private attribute value combinations, hence compromising the privacy of the individuals.

On the basis of this discussion we define a linking attack as follows:

Definition 6 (Linking attack using a model) A linking attack on the privacy of tuples in a table T from domain $A \times B$, using a release M_T, is carried out by

1. Taking a public identifiable database T_{ID} which contains the identities of individuals and their public attributes A.
2. Computing the span for each tuple in T_{ID}.
3. Grouping together all the tuples in T_{ID} that have the same span. This results in sets of tuples, where each set is associated with one span.
4. Listing the possible private attribute value combinations for each span, according to the release M_T.

The tuples that are associated with a span in the third step are now linked to the private attribute value combinations possible for this span according to the fourth step.

For instance, an attacker who knows the identity, gender and marital status of each of the mortgage company's clients in Table 1 can see, by applying the model, that *Donna*, *Emily* and *Fiona* will be classified by means of *leaf 3* (*Female, Married*). This leaf constitutes the span of the relevant tuples. It contains two equivalence classes: one, with a population of 3, of individuals who are identified as *bad* loan risks, and another, with a population of 0, of individuals who are identified as *good* loan risks. Therefore the attacker can link *Donna*, *Emily* and *Fiona* to 3 *bad* loan risk classifications. This example stresses the difference between *anonymity* and *inference* of private data. As mentioned in Sect. 2.1, anonymity depends only on the size of a group of identifiable individuals, regardless of inferred private attribute values. Hence, so long as the k constraint is 3 or less, this information alone does not constitute a k-anonymity breach.

Definition 7 (k-anonymous release) A release M_T is k-anonymous with respect to a table T if a linking attack on the tuples in T using the release M_T will not succeed in linking private data to fewer than k individuals.

Claim 3 A release M_T is k-anonymous with respect to a table T if, for every $x \in T$, either or both of the following hold:

1. $S(x_A) = \{[x]\}$
2. $|S(x_A)_T| \geq k$

Recall that while $S(x_A)_T$ may be different for various tables T, the set of equivalence classes $S(x_A)$ depends only on the model M.

Proof Assume an attacker associated an individual's tuple $(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}}$ with its span $S(x_A)$. We will show that if one of the conditions holds, the attacker cannot compromise the k-anonymity of x. Since this holds for all tuples in T, the release is proven to be k-anonymous.

1. $S(x_A) = \{[x]\}$. Since the equivalence class $[x]$ is the only one in $S(x_A)$, then according to Claim 2, tuples whose span is $S(x_A)$ belong to $[x]$ regardless of their private attribute values. Therefore, no private attribute value can be associated with the span, and the attacker gains no private knowledge from the model in this case. In other words, even if the attacker manages to identify a group of less than k individuals and associate them with $S(x_A)$, no private information will be exposed through this association.
2. $|S(x_A)_T| \geq k$. In this case, the model and the equivalence class populations might reveal to the attacker as much as the exact values of private attributes for tuples in T that belong to equivalence classes in $S(x_A)$. However, since $|S(x_A)_T| \geq k$, the number of individuals (tuples) that can be associated with the span is k or greater.

Note that the first condition pertains to a case that is not mentioned in the original k-anonymity model. This condition characterizes a span that groups tuples by public attributes alone. In the context of tables it is equivalent to suppressing the private attribute values for a set of rows. Clearly there is no privacy risk in this case, even if the set contains less than k rows.

We conclude this section by stressing how the formal definitions relate to the intuitive notion of anonymity that was presented in the beginning of the section. Each equivalence class relates to a subset of tuples which adhere to the same condition on public and private attributes. In that sense, the equivalence class is equivalent to a unique combination of public and private attributes in a row that appears in the private database. Just as a private database does not necessarily adhere to k-anonymity constraints, an equivalence class may contain any number of tuples. However, the spans represent the data as perceived by an attacker whose knowledge is limited to public attributes. Tuples that share the same span have a similar projection on the public domain. k or more tuples that share the same span would result in k or more rows that have the same public attribute values in an equivalent table.

4 Examples

In this section we show how the definition of model k-anonymity given in Sect. 3 can be used to verify whether a given data mining model violates the k-anonymity of individuals whose data was used for its induction.

4.1 k-Anonymity of a decision tree

Assume a mortgage company has the data shown in Table 4 and wishes to release the decision tree in Fig. 2, which clients can use to see whether they are eligible for a loan. Can the company release this decision tree while retaining 3-anonymity for the data in the table?

The *Marital Status* of each individual is common knowledge, and thus a public attribute, while the classification *good/bad* loan risk is private knowledge. We will consider two cases, in which the *Sports Car* attribute can be either public or private.

The decision tree is a function that maps points in the original domain to the leaves of the tree, and inside the leaves, to bins, according to the class value. Hence those bins constitute partitions of the domain—each bin forms an equivalence class and contains all the tuples that are routed to it.

For example, the leaf $l_{\text{Unmarried}}$ contains one *good* loan risk classification, and one *bad* loan risk classification. That is, that leaf contains two bins, distinguished by means of the *Loan Risk* attribute. One tuple from T is routed to the bin labeled *bad*, and one tuple is routed to the bin labeled *good*.

When both *Sports Car* and *Marital Status* are public attributes, the decision tree compromises k-anonymity. For example, the tuple *John* is the only one in the span containing the equivalence classes *good*, *bad* in the leaf

Table 4 Mortgage company data

Name	Marital status	Sports car	Loan risk
Lisa	Unmarried	Yes	Good
John	Married	Yes	Good
Ben	Married	No	Bad
Laura	Married	No	Bad
Robert	Unmarried	Yes	Bad
Anna	Unmarried	No	Bad

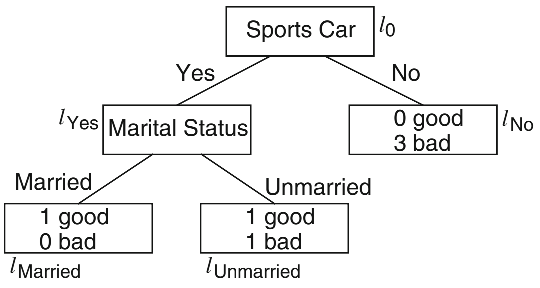

Fig. 2 A k-anonymous decision tree

l_{Married}. Note that in the special case that all the attributes in a decision tree are public and the *Class* attribute is private, the tree is k-anonymous if and only if every leaf contains at least k learning examples or no learning examples at all.

If the *Sports Car* attribute is private, the decision tree implies just two spans: $\{l_{\text{Married/good}}, l_{\text{Married/bad}}, l_{\text{no/good}}, l_{\text{no/bad}}\}$ for *John*, *Ben*, and *Laura* (since the attacker can route these tuples to any of the leaves l_{No}, l_{Married}), and $\{l_{\text{Unmarried/good}}, l_{\text{Unmarried/bad}}, l_{\text{no/good}}, l_{\text{no/bad}}\}$ for *Lisa*, *Robert*, and *Anna* (since the attacker can route these tuples to any of the leaves l_{No}, $l_{\text{Unmarried}}$). As each of these spans contains 3 tuples, the decision tree maintains 3-anonymity.

4.2 Clustering

Assume that a data owner has the data shown in Table 5 and generates the clustering model shown in Fig. 3. Now, he wishes to release the knowledge that his customers form four major groups: One in zip code 11001, comprising customers with various income levels; a second group, of high income customers, living mainly in zip codes 13010 and 14384; a third group, of low income customers, living mainly in zip codes 13010, 14384 and 15012; and a fourth group in zip code 15013, comprising medium and high income customers. This knowledge is released by publication of four centroids, c_1, c_2, c_3, c_4, which represent those groups, and imply a partitioning of the domain into four areas, $C_1, \ldots, C_4$, by assigning the nearest centroid to each point in the domain.

The zip code of each individual is common knowledge, but the income level is private data held only by the data owner. We ask whether the data owner can release this knowledge while retaining 2-anonymity for the data in the table.

Each of the areas C_i implied by the centroids constitutes an equivalence class, and every tuple x is assigned an equivalence class according to its *Zip Code* and *Income* attribute values. The span of a tuple x consists of all the areas that x may belong to when the income corresponding to that tuple is varied across the full range of the data.

The span of *John* and *Cathy* is $\{C_1\}$, because no matter what their income is, any tuple whose zip code is 11001 would be associated (according to Figure 3) with c_1. Because this span has two tuples, it maintains their anonymity.

The span of *Ben*, *Laura* and *William* is $\{C_2, C_3\}$, because unless their income is known, each tuple in the span can be related to either of the two centroids c_2, c_3. This ambiguity maintains the anonymity of these tuples.

The span of *Lisa* is $\{C_3, C_4\}$. It can be seen that this span is not shared by any other tuple; thus, by our definitions, this clustering model compromises 2-anonymity. To see why, consider an attacker who attacks the model with a public table which includes individual names and zip codes. Given the populations of the equivalence classes, the attacker knows that at least one individual has to be related to c_4. The attacker concludes that *Lisa*'s tuple is the only candidate, and thus *Lisa*'s income level is high. Hence *Lisa*'s privacy has been breached.

4.3 k-Anonymity of association rules

Assume that a retailer providing both grocery and pharmaceutical products wishes to provide association rules to an independent marketer. While everyone can see what grocery products a customer purchased (i.e., such items are public knowledge), pharmaceutical products are carried in opaque bags whose content is known only to the customer and the retailer.

After mining the data of some 1,000 customers, the retailer discovers two rules: (*Cherries* $\Longrightarrow$ *Viagra*), with 8.4% support and 75% confidence, and (*Cherries, Birthday Candles* $\Longrightarrow$ *Tylenol*), with 2.4% support and 80%

Table 5 Individuals' data

Name	Zip code	Income
John	11001	98k
Cathy	11001	62k
Ben	13010	36k
Laura	13010	115k
William	14384	44k
Lisa	15013	100k

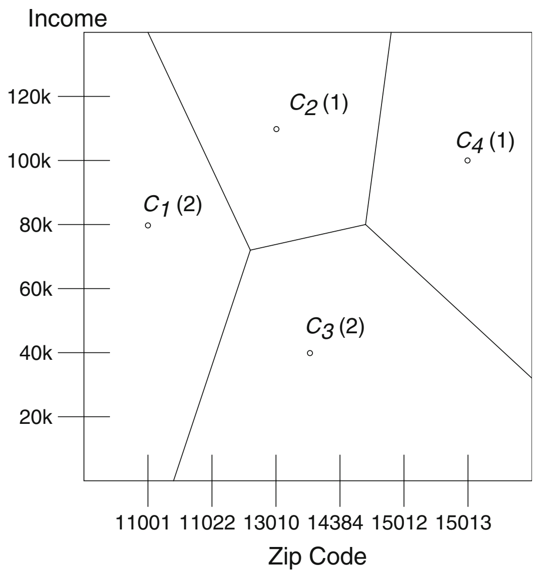

Fig. 3 Clustering model

confidence. Can these rules be transferred to the marketer without compromising customer anonymity?

Given a rule and a tuple, the tuple may contain just a subset of the items on the left-hand side of the rule; all of the items on the left-hand side of the rule; or all of the items on both the left-hand side and the right-hand side of the rule. Applying these three options for each of the two rules results in a model with nine equivalence classes.[1]

By looking at customers' shopping carts, any attacker would be able to separate the customers into three groups, each constituting a span:

S_1: Those who did not buy *Cherries*. The model does not disclose any information about the private items of this group (this span contains a single equivalence class).

S_2: Those who bought both *Cherries* and *Birthday Candles*. Using the confidence and support values of the rules, the attacker can learn private information about their *Viagra* and *Tylenol* purchases (this span contains four equivalence classes).

S_3: Those who bought *Cherries* and did not buy *Birthday Candles*. Using the confidence and support values of the rules, the attacker can learn private information about their *Viagra* purchases (this span contains two equivalence classes).

We can now compute the implied population of every span. There are $\frac{1000 \cdot 0.084}{0.75} = 112$ customers who bought *cherries* (with or without *birthday candles*). There are $\frac{1000 \cdot 0.024}{0.8} = 30$ customers who bought both *cherries* and *birthday candles*, and $112 - 30 = 82$ customers who bought *cherries* and did not buy *birthday candles*. There are $1,000 - 112 = 888$ customers who did not buy *cherries* at all. Therefore, a linking attack would link 888, 30 and 82 individuals to S_1, S_2 and S_3, respectively. Using the confidence of the rule (*Cherries, Birthday Candles* $\Longrightarrow$ *Tylenol*), an attacker can deduce that of the 30 customers linked to S_2, 24 bought Tylenol and 6 did not, which is a breach if the retailer wishes to retain k-anonymity for $k > 30$.

We conclude that if the objective of the retailer is to retain 30-anonymity, then it can safely release both rules. However, if the retailer wishes to retain higher anonymity, the second rule cannot be released because it would allow an attacker to link a small group of customers to the purchase of *Tylenol*.

[1]In fact there are only seven equivalence classes, since the rules overlap: A tuple that does not contain items from the left-hand side of the first rule ('no cherries') cannot be classified as containing the items on the left-hand side or on both sides of the second rule.

5 *k*-Anonymity preserving data mining algorithms

In the previous section we used our definition of k-anonymity to test whether an existing model violates the anonymity of individuals. However, it is very probable that the output of a data mining algorithm used on non-anonymized data would cause a breach of anonymity. Hence the need for techniques to produce models which inherently maintain a given anonymity constraint. We now demonstrate data mining algorithms which guarantee that only k-anonymous models will be produced.

5.1 Inducing k-anonymized decision trees

We present an algorithm that generates k-anonymous decision trees, given a set of tuples T, assuming $|T| > k$. The outline is given in Algorithm 1. We accompany the description of the algorithm with an illustration of a 3-anonymous decision tree induction, given in Fig. 4. It shows an execution of the algorithm using the data in Table 4 as input. *Marital Status* is a public attribute; *Sports Car* and *Loan risk* are private attributes. The result of the execution is the decision tree in Fig. 2.

The algorithm is based on concepts similar to those of the well-known ID3 decision tree induction algorithm [30]. The algorithm begins with a tree consisting of just the root and a set of learning examples associated with the root. Then it follows a hill climbing heuristic that splits the set of learning examples according to the value the examples have for a selected attribute. Of all the given nodes and attributes by which it can split the data, the algorithm selects the one which yields the highest

Algorithm 1 Inducing k-anonymous decision tree

```
1: procedure MakeTree(T,A,$k$)
       ▷ T – dataset, A – list of attributes, $k$ – anonymity parameter
2:     $r \leftarrow$ root node.
3:     $candList \leftarrow \{(a,r) : a \in A\}$
4:     while $candList$ contains candidates with positive gain do
5:         $bestCand \leftarrow$ candidate from $candList$ with highest gain.
6:         if $bestCand$ maintains $k$-anonymity then
7:             Apply the split and generate new nodes $N$.
8:             Remove candidates with the split node from $candList$.
9:             $candList \leftarrow candList \cup \{(a,n) : a \in A, n \in N\}$.
10:        else
11:            remove $bestCand$ from $candList$.
12:        end if
13:    end while
14:    return generated tree.
15: end procedure
```

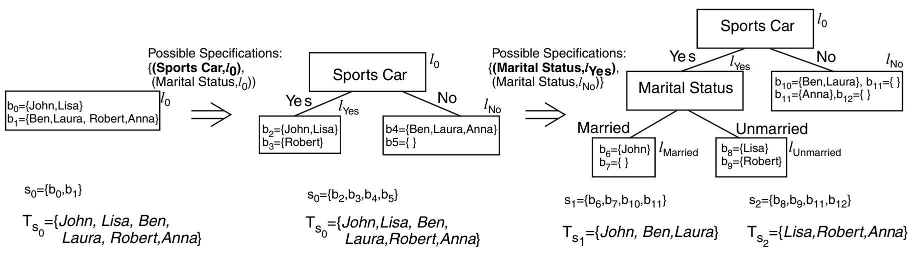

Fig. 4 Inducing a 3-anonymous decision tree

gain (for a specific gain function—e.g., Information Gain or the Gini Index), provided that such a split would not cause a breach of k-anonymity. Note that unlike the ID3 algorithm, our algorithm does not use recursion; we consider instead all the splitting possibilities of all the leaves in a single queue, ordered by their gain. That is because splitting leaves might affect the k-anonymity of tuples in other leaves.

For simplicity, we embed generalization in the process by considering each possible generalization of an attribute as an independent attribute. Alternatively, e.g., for continuous attributes, we can start with attributes at their lowest generalization level. Whenever a candidate compromises anonymity and is removed from the candidate list, we insert into the candidate list a new candidate with a generalized version of the attribute. In that case, when generating candidates for a new node, we should consider attributes at their lowest generalization level, even if they were discarded by an ancestor node.

To decide whether a proposed split in line 6 would breach k-anonymity, the algorithm maintains a list of all tuples, partitioned to groups T_s according to the span s they belong to. Additionally, at every bin on every leaf, the span containing that bin $s(b)$ is stored. Lastly, for every span there is a flag indicating whether it is pointed to by a single bin or by multiple bins.

Initially, in line 2, the following conditions hold:

– the only leaf is the root;
– there are as many bins as class values;
– there is just one span if the class is private;
– there are as many spans as class values if the class is public.

If the class is private, the population of the single span is T and its flag is set to *multiple*. If it is public, the population of every span is the portion of T which has the respective class value, and the flag of every span is set to *single*.

In Fig. 4, we begin with the root node l_0, which contains two bins, one for each class value. As the class value is private, only one span s_0 is created: it contains the two bins and its flag is set to *multiple*. Ts_0, the population of s_0, is comprised of all the tuples.

When a leaf is split, all of its bins are also split. The algorithm updates the data structure as follows:

– If the splitting attribute is public, then the spans are split as well, and tuples in T_s are distributed among them according to the value of the splitting attribute. Every new bin will point to the corresponding span, and the flag of every new span will inherit the value of the old one.
– If the splitting attribute is private, then every new bin will inherit the old span. The flag of that span will be set to *multiple*.

If splitting a leaf results in a span with population smaller than k and its flag set to *multiple*, k-anonymity will be violated. In that case the splitting is rolled back and the algorithm proceeds to consider the attribute with the next largest gain.

In the example, there are two candidates for splitting the root node: the *Sports Car* attribute and the *Marital Status* attribute. The first one is chosen due to higher information gain. Two new leaves are formed, l_{Yes} and l_{No}, and the bins are split among them according to the chosen attribute. Since the *Sports Car* attribute is private, an attacker will not be able to use this split to distinguish between tuples, and hence the same span s_0 is maintained, with the same population of size >3 (hence 3-anonymous). There are two remaining candidates. Splitting l_{No} with *MaritalStatus* is discarded due to zero information gain. The node l_{Yes} is split using the public *MaritalStatus*. As a consequence, all the bins in s_0 are also split according to the attribute, and s_0 is split to two new spans, s_1 and s_2, each with a population of three tuples, hence maintaining 3-anonymity.

5.2 Inducing k-anonymized clusters

We present an algorithm that generates k-anonymous clusters, given a set of tuples T, assuming $|T| > k$. The algorithm is based on a top-down approach to clustering [35].

The algorithm starts by constructing a minimal spanning tree (MST) of the data. This tree represents a single equivalence class, and therefore a single span. Then, in

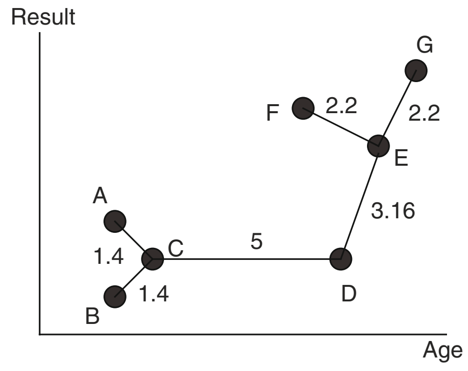

Fig. 5 Inducing k-anonymous clusters

consecutive steps, the longest MST edges are deleted to generate clusters. Whenever an edge is deleted, a cluster (equivalence class) C_i is split into two clusters (two equivalence classes), C_{i_1} and C_{i_2}. As a consequence, every span $M = \{C_1, \ldots, C_i, \ldots, C_m\}$ that contained this equivalence class is now split into three spans:

1. $S_1 = \{C_1, \ldots, C_{i_1}, \ldots, C_m\}$, containing the points from C_i which, according to the public attribute values, may belong only to C_{i_1};
2. $S_2 = \{C_1, \ldots, C_{i_2}, \ldots, C_m\}$, containing the points from C_i which, according to the public attribute values, may belong only to C_{i_2};
3. $S_3 = \{C_1, \ldots, C_{i_1}, C_{i_2}, \ldots, C_m\}$, containing the points from C_i which, according to the public attribute values, may belong either to C_{i_1} and C_{i_2}.

The points that belonged to M are now split between the three spans. If we can link to each of the new spans at least k points, or no points at all, then the split maintains anonymity. Otherwise, the split is not performed. Edges are deleted iteratively in each new cluster, until no split that would maintain anonymity can be performed. When this point is reached, the algorithm concludes. The algorithm can also be terminated at any earlier point, when the data owner decides that enough clusters have been formed.

To see how the algorithm is executed, assume a data domain that contains two attributes. The attribute *age* is public, while the attribute *result*, indicating a result of a medical examination, is private. Figure 5 shows several points in the domain, and an MST that was constructed over these points.

The algorithm proceeds as follows: At first, the MST forms a single cluster (equivalence class) C_1, containing all the points, and a single span $S_1 = \{C_1\}$. Then the edge CD, which is the longest, is removed. Two clusters form as a result: $C_2 = \{A, B, C\}$ and $C_3 = \{D, E, F, G\}$. Consequently, we get three spans: $S_2 = \{C_2\}$, to which the points A, B, C are linked; $S_3 = \{C_3\}$, to which the points D, E, F, G are linked; and $S_4 = \{C_2, C_3\}$, to which no point is linked, and can therefore be ignored. In C_3, the longest edge is DE. Removing it will split the cluster into $C_4 = \{D\}$ and $C_5 = \{E, F, G\}$. Then the span S_3, which contains the split cluster C_3, is split into three spans: $S_5 = \{C_4\}$, to which no point is linked; $S_6 = \{C_5\}$, to which no point is linked; and $S_7 = \{C_4, C_5\}$, to which the points D, E, F, G are linked. Note that although point D is the only one in equivalence class C_4, this does not compromise k-anonymity, because the public attributes do not reveal enough information to distinguish it from the points E, F, G in cluster C_5. Although the algorithm may continue to check other possible splits, it can be terminated at this point, after forming three clusters.

6 Experimental evaluation

In this section we provide some experimental evidence to demonstrate the usefulness of the model we presented. We focus on the decision tree classification problem and present results based on the decision tree algorithm from Sect. 5.1.

To conduct our experiments we use a straightforward implementation of the algorithm, based on the Weka package [41]. We use as a benchmark the Adult database from the UC Irvine machine learning repository [13], which contains census data, and has become a commonly used benchmark for k-anonymity. The data set has 6 continuous attributes and 8 categorial attributes. We use the income level as the class attribute, with two possible income levels, ≤ 50 K or > 50 K. After records with missing values have been removed, there are 30,162 records for training and 15,060 records for testing (of which 24.5% are classified > 50 K). For the categorial attributes we use the same hierarchies described in [18]. We dropped the continuous attributes because of ID3 limitations. The experiment was performed on a 3.0GHz Pentium IV processor with 512 MB memory.

The anonymized ID3 algorithm uses the training data to induce an anonymous decision tree. Then the test data (in a non-anonymized form) is classified using the anonymized tree. For all values of k the decision tree induction took less than 6 seconds.

6.1 Accuracy versus Anonymity Tradeoffs in ID3

The introduction of privacy constraints forces loss of information. As a consequence, a classifier is induced with less accurate data and its accuracy is expected to decrease. Our first goal is to assess how our method impacts this tradeoff between classification accuracy and the privacy constraint.

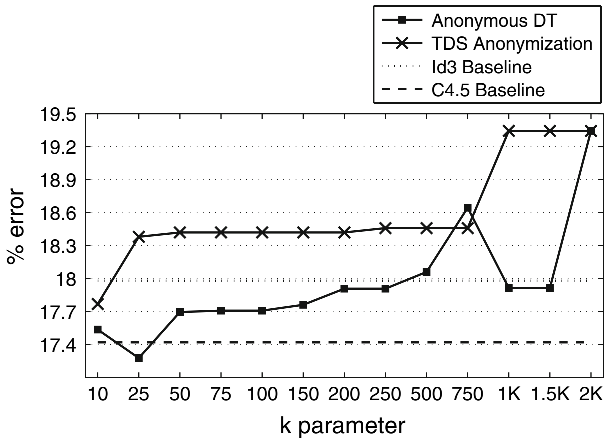

Fig. 6 Classification error versus k parameter

Figure 6 shows the classification error of the anonymous ID3 for various k parameters. We provide an ID3 baseline, as well as a C4.5 baseline (obtained using 0.27 confidence factor), to contrast the pruning affect of k-anonymity. In spite of the anonymity constraint, the classifier maintains good accuracy. At $k = 750$ there is a local optimum when the root node is split using the *Relationship* attribute at its lowest generalization level. At $k = 1{,}000$ this attribute is discarded since it compromises anonymity, and instead the *Marital Status* attribute is chosen at its second lowest generalization level, yielding better classification.

We compare the classification error with the one obtained using the top-down specialization (TDS) algorithm presented in [18] on the same data set and the same attributes and taxonomy trees. The TDS algorithm starts with the topmost generalization level and chooses, in every iteration, the best specialization. The best specialization is determined according to a metric that measures the information gain for each unit of anonymity loss. The generalization obtained by the algorithm is then used to anonymize all the tuples. We compare our anonymous decision tree with an ID3 tree induced with the anonymized TDS output. The results obtained using TDS also appear in Fig. 6. In contrast to the TDS algorithm, our algorithm can apply different generalizations on different groups of tuples, and it achieves an average reduction of 0.6% in classification error with respect to TDS.

6.2 Model-based k-Anonymization

In Sect. 3 we discussed the concept of equivalence between data mining models and their table representation. Based on this concept, we can use data mining techniques to anonymize data. Each span defines an anonymization for a group of at least k tuples.

Table 6 Classification metric comparison

k	k-optimal	Anonymous-DT
10	5,230–5280	5,198
25	5,280–5330	5,273
50	5,350–5410	5,379
100	5,460	5,439

When the data owner knows in advance which technique will be used to mine the data, it is possible to anonymize the data using a matching technique. This kind of anonymization would be very similar to embedding the anonymization within the data mining process. However, when the algorithm for analysis is not known in advance, would a data-mining-based anonymization algorithm still be useful? To answer this question, we next assess the value of the anonymous decision tree as an anonymization technique for use with other classification algorithms.

To make this assessment, we measured the classification metric (originally proposed in [21]) for the induced decision trees. This metric was also used in [7] for optimizing anonymization for classification purposes. In our terminology, the classification metric assigns a penalty 1 to every tuple x that does not belong to the majority class of $S(x)$, $CM = \sum_{\forall S} |\text{Minority}(S)|$. We compare our results with the k-Optimal algorithm presented in [7], which searches the solution domain to find an optimal anonymization with respect to a given metric. We discarded the *Relationship* attribute, since it is not used in [7]. Note also that we do not make use of the *Age* attribute, which is used in [7]. This puts our algorithm at a disadvantage. [7] reports several CM values, depending on the partitioning imposed on the *Age* attribute and the limit on number of suppressions allowed (our algorithm makes no use of suppressions at all). We present in Table 6 the ranges of CM values reported in [7] alongside the CM results achieved by our algorithm. Our algorithm obtains similar (sometimes superior) CM results in a shorter runtime.

Successful competition with optimal single-dimension anonymizations using multi-dimensional anonymizations has already been discussed in [26]. However, our results give rise to an additional observation: Intuitively it may seem that using a specific data mining algorithm to generalize data would “over-fit” the anonymization scheme to the specific algorithm, decreasing the ability to successfully mine the data using other algorithms. However, the CM results presented above suggest that this kind of anonymization may be at least as useful as metric-driven (and algorithm oblivious) anonymization.

6.3 Privacy risks and ℓ-diversity

As mentioned in Sect. 2.1, k-anonymity makes no restriction regarding the private attribute values. As a consequence, it is possible that a k-anonymous model would allow the attacker a complete inference of these values. In this section, our goal is to assess how many individuals are prone to immediate inference attacks and investigate whether such inference can be thwarted using the ℓ-diversity model [28].

Specifically, we look at the number of individuals (learning examples) for whom an attacker may infer the class attribute value with full certainty. This is done by considering all the spans for which all the tuples share the same class, and counting the number of tuples associated with these spans. Figure 7 shows the percentage of tuples exposed to such inference, as a function of the parameter k. Avoiding this kind of inference completely requires high values of k, and even in those cases the attacker may still be able to infer attribute values with high probability.

The ℓ-diversity model suggests solving this problem by altering the privacy constraint to one that requires a certain amount of diversity in class values for every group of identifiable tuples. For example, *entropy ℓ-diversity* is maintained when the entropy of the class values for every such group exceeds a threshold value $log(\ell)$.

We altered our algorithm to enforce the entropy ℓ-diversity constraint: instead of checking the number of tuples associated with each span, we calculated the class entropy and compared it to the threshold $\log(\ell)$, ruling out splits in the tree that violate this constraint. Before presenting the results, we make some observations about entropy ℓ-diversity. In the given data set there are two class values. This means that the best level of diversity we can hope for is entropy 2-diversity, when there is equal chance for each class value, in which case we have no classification ability. Therefore, in this context, the parameters for k-anonymity and ℓ-diversity are not comparable.

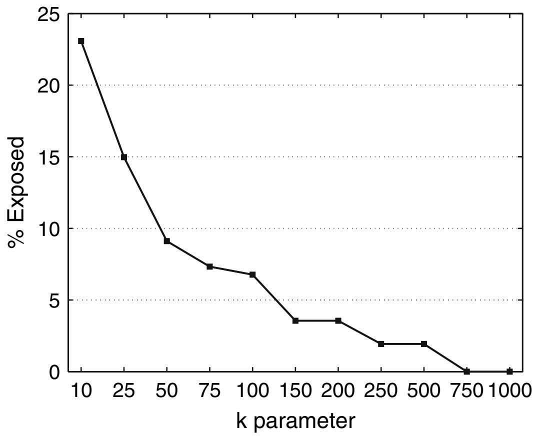

Fig. 7 Percentage of Exposed tuples

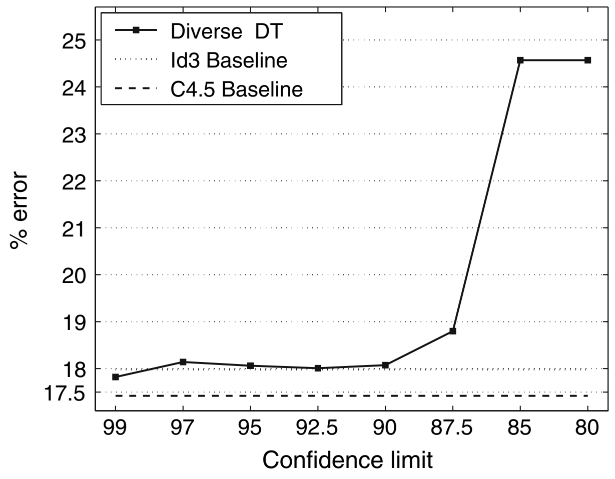

Fig. 8 Confidence level versus accuracy

However, when we have two class values and $\ell < 2$, entropy ℓ-diversity allows us to limit the attacker's confidence in inference attacks. For example, to deny the attacker the ability to infer a class value with confidence $> 85\%$, we should keep the entropy higher than $-0.85 \times \log 0.85 - 0.15 \times \log 0.15 = 0.61$. This amounts to applying entropy ℓ-diversity with $\ell = 1.526$ ($\log 1.526 = 0.61$). Based on this, Fig. 8 displays the tradeoff between the confidence limit imposed on the attacker and the accuracy of the induced decision tree. According to our results, so long as the confidence limit is high enough, the ℓ-diversity constraint allows the induction of decision trees without a significant accuracy penalty. The lowest achievable confidence level is 75.1%, as it pertains to the class distribution in the root node. Moreover, every split of the root node will result in a node with confidence $>$ 85%. Therefore, a confidence limit of 85% or lower prohibits the induction of a useful decision tree.

7 Related work

The problem of k-anonymity has been addressed in many papers. The first methods presented for k-anonymization were bottom-up, relying on generalization and suppression of the input tuples [32,33,40]. Heuristic methods for k-anonymization that guarantee optimal k-anonymity were suggested in [7,25]. Iyengar [21] suggested a metric for k-anonymizing data used in classification problems, and used genetic algorithms for the anonymization process. A top-down approach, suggested in [18], preserves data patterns used for decision tree classification. Another top down approach, suggested in [8] utilizes usage metrics to bound generalization and guide the anonymization process. When the above methods are compared to specific implementations of ours (such as the one described in Sect. 5.1), several differences are revealed. First, while all these methods anonymize an attribute across all tuples, ours selectively anonymizes attributes for groups of tuples. Second, our method is general and thus can be easily adjusted for *any* data mining task. For example, one could think of applying our method in one way for classification using decision trees and in another for classification using Bayesian classifiers. In both respects, using our method is expected to yield better data mining results, as was demonstrated in Sect. 6 for decision tree induction.

A recent independent work [26] discusses multi-dimensional global recoding techniques for anonymization. Anonymity is achieved by mapping the domains of the quasi-identifier attributes to generalized or altered values, such that each mapping may depend on the combination of values over several dimensions. The authors suggest that multi-dimensional recoding may lend itself to creating anonymizations that are useful for building data mining models. Indeed, the methods we presented in this paper can be classified as multi-dimensional global recoding techniques, and complement the aforementioned work. Another multi-dimensional approach is presented in [3]. The authors provide an $O(k)$-approximation algorithm for k-anonymity, using a graph representation, and provide improved approximation algorithms for $k = 2$ and $k = 3$. Their approximation strives to minimize the cost of anonymization, determined by the number of entries generalized and the level of anonymization. One drawback of applying multi-dimensional global recoding before mining data is the difficulty of using the anonymized results as input for a data mining algorithm. For example, determining the information gain of an attribute may not be trivial when the input tuples are generalized to different levels. Our approach circumvents this difficulty by embedding the anonymization within the data mining process, thus allowing the data mining algorithm access to the non-anonymized data.

Embedding k-anonymity in data mining algorithms was discussed in [6] in the context of pattern discovery. The authors do not distinguish between private and public items, and focus on identifying patterns that apply for fewer than k transactions. The authors present an algorithm for detecting inference channels in released sets of itemsets. Although this algorithm is of exponential complexity, they suggest an optimization that allows running time to be reduced by an order of magnitude. A subsequent work [5] shows how to apply this technique to assure anonymous output of frequent itemset mining. In comparison, our approach allows breaches of k-anonymity to be detected and k-anonymization to be embedded in a broader range of data mining models. We intend to further explore the implications of our approach on itemset mining in future research.

Several recent works suggest new privacy definitions that can be used to overcome the vulnerability of k-anonymity with respect to data diversity. Wang et al. [39] offer a template-based approach for defining privacy. According to their method, a data owner can define risky inference channels and prevent learning of specific private attribute values, while maintaining the usefulness of the data for classification. This kind of privacy is attained by selective suppression of attribute values. [28] presents the ℓ-diversity principle: Every group of individuals that can be isolated by an attacker should contain at least ℓ "well-represented" values for a sensitive attribute. As noted in [28], k-anonymization methods can be easily altered to provide ℓ-diversity. We showed in Sect. 6 that our method can also be applied to ℓ-diversification by adding private value restrictions on spans. Our definitions can be easily augmented with any further restriction on private attributes values, such as those presented in [39]. Kantarcioğlu et al. [23] suggest another definition for privacy of data mining results, according to the ability of the attacker to infer private data using a released "black box" classifier. While this approach constitutes a solution to the inference vulnerability of k-anonymity, it is not clear how to apply it to data mining algorithms such that their output is guaranteed to satisfy privacy definitions.

A different approach for privacy in data mining suggests that data mining should be performed on perturbed data [2,9,11,12,15,16,23]. This approach is applied mainly in the context of statistical databases.

Cryptographic methods were proposed for privacy-preserving data mining in multiparty settings [14,19, 22,27,37]. These methods deal with the preservation of privacy in the *process* of data mining and are thus complementary to our work, which deals with the privacy of the output.

We refer the interested reader to [38] for further discussion of privacy preserving data mining.

8 Conclusions

Traditionally, the data owner would anonymize the data and then release it. Often, a researcher would then take the released data and mine it to extract some knowledge.

However, the process of anonymization is oblivious to any future analysis that would be carried out on the data. Therefore, during anonymization, attributes critical for the analysis may be suppressed whereas those that are not suppressed may turn out to be irrelevant. When there are many public attributes the problem is even more difficult, due to the *curse of dimensionality* [1]. In that case, since the data points are distributed sparsely, the process of *k*-anonymization reduces the effectiveness of data mining algorithms on the anonymized data and renders privacy preservation impractical.

Using data mining techniques as a basis for *k*-anonymization has two major benefits, which arise from the fact that different data mining techniques consider different representations of data. First, such anonymization algorithms are optimized to preserve specific data patterns according to the underlying data mining technique. While this approach is more appealing when the data owner knows in advance which tool will be used to mine the data, our experiments show that these anonymizations may also be adequate when this is not the case. Second, as illustrated in sect. 3, anonymization algorithms based on data mining techniques may apply different generalizations for several groups of tuples rather than the same generalization for all tuples. In this way, it may be possible to retain more useful information. This kind of anonymization, however, has its downsides, one of which is that using different generalization levels for different tuples requires that the data mining algorithms be adapted. Therefore, we believe that this model will be particularly useful when the anonymity constraints are embedded within the data mining process, so that the data mining algorithm has access to the non-anonymized data.

To harness the power of data mining, our work proposes extended definitions of *k*-anonymity that allow the anonymity provided by a data mining model to be analyzed. Data owners can thus exchange models which retain the anonymity of their clients. Researchers looking for new anonymization techniques can take advantage of efficient data mining algorithms: they can use the extended definitions to analyze and maintain the anonymity of the resulting models, and then use the anonymity preserving models as generalization functions. Lastly, data miners can use the definitions to create algorithms guaranteed to produce anonymous models.

References

1. Aggarwal, C.C.: On *k*-anonymity and the curse of dimensionality. In: VLDB, pp. 901–909 (2005)
2. Aggarwal, C.C., Yu, P.S.: A condensation approach to privacy preserving data mining. In: EDBT, pp. 183–199 (2004)
3. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for *k*-anonymity. J. Privacy Technol. (JOPT) (2005)
4. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the ACM SIGMOD'00, pp. 439–450. Dallas, Texas, (2000)
5. Atzori, M., Bonchi, F., Giannotti, F., Pedreschi, D.: Blocking anonymity threats raised by frequent itemset mining. In: ICDM, pp. 561–564 (2005)
6. Atzori, M., Bonchi, F., Giannotti, F., Pedreschi, D.: *k*-anonymous patterns. In: PKDD, pp. 10–21 (2005)
7. Bayardo, R.J.: Jr. Agrawal, R.: Data privacy through optimal k-anonymization. In: ICDE, pp. 217–228 (2005)
8. Bertino, E., Chin Ooi, B., Yang, Y., Deng, R.H.: Privacy and ownership preserving of outsourced medical data. In: ICDE, pp. 521–532 (2005)
9. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the SuLQ framework. In: Proceedings of PODS'05, pp. 128–138. ACM Press, New York (2005)
10. Electronic Privacy Information Center: ~~Total~~ "terrorism" information awareness (TIA). http://www.epic.org/privacy/profiling/tia/.
11. Chawla, S., Dwork, C., McSherry, F., Smith, A., Wee, H.: Toward privacy in public databases. In: Theory of Cryptography Conference, pp. 363–385 (2005)
12. Dinur, I., Nissim, K.: Revealing information while preserving privacy. In: Proceedings of PODS'03, pp. 202–210 (2003)
13. Blake, C.L., Newman, D.J., Hettich, S., Merz, C.J.: UCI repository of machine learning databases (1998)
14. Du, W., Zhan, Z.: Building decision tree classifier on private data. In: Proceedings of CRPITS'14, pp. 1–8. Australian Computer Society, Inc, Darlinghurst (2002)
15. Dwork, C., Nissim, K.: Privacy-preserving data mining on vertically partitioned databases. In: Proceedings of CRYPTO'04 (2004)
16. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: Proceedings of PODS'03, pp. 211–222. San Diego, California, USA, 9–12 June 2003
17. Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: Proceedings of ACM SIGKDD'02, pp. 217–228. Canada (2002)
18. Fung, B.C.M., Wang, K., Yu, P.S.: Top–down specialization for information and privacy preservation. In: Proceedings of ICDE'05, Tokyo (2005)
19. Gilburd, B., Schuster, A., Wolff, R.: k-ttp: a new privacy model for large-scale distributed environments. In: Proceedings of ACM SIGKDD'04, pp. 563–568 (2004)
20. Huang, Z., Du, W., Chen, B.: Deriving private information from randomized data. In: Proc. of ACM SIGMOD'05 (2005)
21. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proceedings of ACM SIGKDD'02, pp. 279–288 (2002)
22. Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned data. In: Proceedings of DKMD'02, (2002)
23. Kantarcioğlu, M., Jin, J., Clifton, C.: When do data mining results violate privacy? In: Proceedings of ACM SIGKDD'04, pp. 599–604. ACM Press, New York (2004)
24. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: Proceddings of ICDM'03, pp. 99. IEEE Computer Society, Washington (2003)
25. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: efficient full-domain *k*-anonymity. In: Proceedings of SIGMOD'05, pp. 49–60. ACM Press, New York (2005)

26. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proceedings of ICDE (2006)
27. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Proceedings of CRYPTO'04, pp. 36–54. Springer (2000)
28. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-diversity: privacy beyond k-anonymity. In: Proceedings of ICDE (2006)
29. Meyerson, A., Williams, R.: General k-anonymization is hard. In: Proceedings of PODS'04 (2003)
30. Ross Quinlan, J.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
31. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans Know Data Eng. **13**(6), (2001), 1041–4347. DOI 10.1109/69.971193
32. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In: Technical Report SRI-CSL-98-04. CS Laboratory, SRI International (1998)
33. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(5), 571–588 (2002)
34. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl. Based Syst. **10**(5), 557–570 (2002)
35. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans. Comput. C- **20**, 68–86 (1971)
36. US Dept. of HHS: Standards for privacy of individually identifiable health information; final rule (2002)
37. Vaidya, J., Clifton, C.: Privacy preserving association rule mining in vertically partitioned data. In: Proceedings of ACM SIGKDD '02, Edmonton, (2002)
38. Verykios, V.S., Bertino, E., Nai Fovino, I., Provenza, L.P., Saygin, Y., Theodoridis, Y.: State-of-the-art in privacy preserving data mining. SIGMOD Rec. **33**(1), 50–57 (2004)
39. Wang, K., Fung, B.C.M., Yu, P.S.: Template-based privacy preservation in classification problems. In: ICDM, pp. 466–473 (2005)
40. Wang, K., Yu, P.S., Chakraborty, S.: Bottom-up generalization: a data mining solution to privacy protection. In: ICDM, pp. 249–256 (2004)
41. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques, 2nd ed. Morgan Kaufmann, San Francisco, (2005)
42. Zhong, S., Yang, Z., Wright, R.N.: Privacy-preserving classification of customer data without loss of accuracy. In: SIAM International Conference on Data Mining (SDM), Newport Beach (2005)