k-Anonymous Decision Tree Induction

Arik Friedman, Assaf Schuster, and Ran Wolff

Technion – Israel Institute of Technology
Computer Science Dept.
{arikf, assaf, ranw}@cs.technion.ac.il

Abstract. In this paper we explore an approach to privacy preserving data mining that relies on the k-anonymity model. The k-anonymity model guarantees that no private information in a table can be linked to a group of less than k individuals. We suggest extended definitions of k-anonymity that allow the k-anonymity of a data mining model to be determined. Using these definitions, we present decision tree induction algorithms that are guaranteed to maintain k-anonymity of the learning examples. Experiments show that embedding anonymization within the decision tree induction process provides better accuracy than anonymizing the data first and inducing the tree later.

Keywords: k-anonymity, privacy preserving data mining, decision trees.

1 Introduction

In recent years, the effectiveness of data mining tools in revealing the knowledge locked within huge databases has raised concerns about its impact on the privacy of individuals. Two main approaches to privacy preserving data mining were suggested. The *data transformation approach* – e.g., [1] – tries to modify the data so as to hide the sensitive information while retaining interesting patterns. The *cryptographic approach* – e.g., [2,3,4] – uses cryptographic tools to prevent information leakage during the computation of the data mining model. The latter approach only applies to distributed data mining and does not prevent leakage due to the model itself (see, for example, [5]).

k-anonymity – a definition for privacy that was conceived in the context of databases – has come a long way in the public arena. Roughly speaking, k-anonymity provides a "blend into the crowd" approach to privacy. It assumes that the owner of a data table can separate the columns into *public* ones (*quasi-identifiers*) and *private* ones. Public columns may appear in external tables, and thus be available to an attacker. Private columns contain data which is not available in external tables and needs to be protected. The guarantee provided by k-anonymity is that an attacker would not be able to link private information to groups of less than k individuals. This is enforced by making certain that every combination of public attribute values appears in at least k rows of the released table, or in no row at all. k-anonymity is accepted today by both legislators and corporations, and is considered to provide the kind of privacy required by legislation such as the HIPAA [6].

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): PKDD 2006, LNAI 4213, pp. 151–162, 2006.

Table anonymization is NP-Hard [7]. Thus, heuristic efficient anonymization of tables is the concern of most work in the area [8,9,10,11,12,13]. Specific care is given to preserving as much of the original data as possible. Interestingly, some of this work deals with preserving data which would be useful should the table be data mined following its release [9,10,11]. Data mining is envisioned as the main target application of released data.

This paper takes a direct approach for the combination of k-anonymity and data mining. Rather than asking how data can be anonymized so that it is useful for data mining, we ask how can data be mined so that the resulting model is guaranteed to provide k-anonymity. We specifically discuss this question in the context of decision tree induction. In this context, anonymity is at risk when the decision tree overfits a small set of learning examples and allows the attacker to predict their private attribute values. We describe a decision tree induction algorithm whose output is guaranteed not to compromise the k-anonymity of the data from which it was induced. An independent work [14] presents a similar concept in the context of itemset mining. However, that work does not differentiate public attributes from private attributes, and is limited to binary attributes. In addition, anonymity is achieved by postprocessing of the data mining output, while we suggest an integration of the two processes.

Our approach is superior to existing methods (e.g., [9,10,11]), which guarantee k-anonymity of a data mining model by building it from a k-anonymized table. For the sake of efficiency, these methods generalize attributes homogenously over all the tuples. This kind of anonymization was termed *single-dimension recoding* [15]. Using our method, however, attributes can be generalized differently in each tuple, depending on other attribute values. This kind of anonymization was termed *multi-dimensional recoding*. Furthermore, in the existing methods, heuristic cost metrics are the driving force. For example, a classification metric, which is in essence the classification error over the entire data, may be used. Such metrics are not necessarily optimal for a specific data mining task. We show that a decision tree induced using our method is usually more accurate than that induced by existing methods. Needless to say, both decision trees provide the same level of anonymity.

This paper makes the following contributions:

- It suggests extended definitions of k-anonymity, which allow the k-anonymity of a data mining model with respect to the learning examples to be determined.
- It demonstrates how the definitions of k-anonymity can be applied to determine the anonymity of a decision tree.
- It presents a decision tree induction algorithm which guarantees k-anonymous output and which performs better than existing methods in terms of accuracy on standard benchmarks.

The organization of this paper is as follows: Section 2 outlines the extended definitions of k-anonymity of data mining models. Section 3 demonstrates

how the definitions can be incorporated within decision tree induction algorithms to guarantee k-anonymous output. Section 4 compares this new algorithm experimentally to previous work. Conclusions are drawn in Sect. 5.

2 Extending k-Anonymity to Models

We start by extending the definition of k-anonymity beyond the release of tables. Just as in the original k-anonymity model, we assume that the data owner can determine which attributes are known to a potential attacker and can be used to identify individuals, and which attributes are private knowledge. Additionally, without loss of generality, we assume that each tuple in the database pertains to a different individual.

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

We denote $A = A_1 \times \ldots \times A_k$ 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 $[x] = \{y \in D : y \equiv x\}$. 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.

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., $p_T([x]) = |T \bigcap [x]|$.*

In our terminology, a decision tree model is a function that assigns bins to tuples in D. 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. A release of a decision tree includes the partition into bins, as well as the number of learning examples that populate each bin.

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. In other words, different definitions of p_T would reveal different private information on the same groups of tuples.

We now turn to see how the database and the model are perceived by an attacker. One of the fundamental assumptions of the k-anonymity model is about the data available to the attacker.

Definition 4 (A Public Identifiable Database). *A public identifiable database $T_{\mathrm{ID}} = \{(\mathrm{id}_x, x_A) : x \in T\}$ 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}. Consider a tuple $(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}}$. As each equivalence class in M may rely on both private and public attributes, the attacker, could he associate x_A with the correct equivalence class, could then infer which private attribute values are possible for x according to this equivalence class. However, depending on the unknown private attribute values of x, there might be a number of possible equivalence classes $[x]$ to which an attacker can associate x_A. 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 and 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 $S(a)$.*

For example, in a decision tree, the span of a tuple is the set of bins to which the tuple can be routed when any combination of private attribute values is possible for it. 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 of M. Just as associating an individual with an equivalence class would allow private attribute values to be inferred, knowing the values of p_T for each equivalence class in $S(a)$ allows the possible private attribute value combinations for the tuples in $S(a)_{T_{\mathrm{ID}}}$ to be constrained, hence compromising the privacy of the individuals.

Definition 6 (Linking Attack Using a Release). *A linking attack using a release M_T and a public identifiable database T_{ID} is performed by grouping tuples in T_{ID} according to their spans. Each group of tuples is then linked to the list of possible private attribute value combinations, according to M_T.*

Definition 7 (k-Anonymous Model). *A model M is k-anonymous with respect to a private table T if a linking attack on the tuples in T_{ID} using the release*

M_T will not succeed in linking private data to fewer than k individuals. In other words, a model M is k-anonymous with respect to T if, for every $(\mathrm{id}_x, x_A) \in T_{\mathrm{ID}}$, $|S(x_A)_{T_{\mathrm{ID}}}| \geq k$.

For example, consider the decision tree in Fig. 1. The decision tree was formed using the data in Table 1. The *Marital Status* attribute is public, while the *Sports Car* and *Loan Risk* attributes are private. The *Loan Risk* attribute is the class attribute. The decision tree includes six bins, each with its population denoted in the tree. The tuples of *Anna* and *Ben* belong to the same bin, because of the *Sports Car* and *Loan Risk* attributes. The model ignores the value of the *Marital Status* attribute for these tuples. On the other hand, an attacker, who has no access to the *Sports Car* attribute values, is forced to consider routing *Anna*'s tuple to the leaf $l_{\mathrm{Unmarried}}$, and routing *Ben*'s tuple to the leaf l_{Married}, in addition to routing each of them to the leaf l_{No}. Therefore, the decision tree implies two spans: $\{l_{\mathrm{Married/good}}, l_{\mathrm{Married/bad}}, l_{\mathrm{no/good}}, l_{\mathrm{no/bad}}\}$ for *John*, *Ben*, and *Laura*, and $\{l_{\mathrm{Unmarried/good}}, l_{\mathrm{Unmarried/bad}}, l_{\mathrm{no/good}}, l_{\mathrm{no/bad}}\}$ for *Lisa*, *Robert*, and *Anna*. The attacker can use the populations to induce the distribution of class attribute values within each span, but he cannot know, for example, which of the three tuples in the first span belongs to $l_{\mathrm{Married/good}}$. As each span contains three tuples, the model is 3-anonymous with respect to Table 1.

Table 1. 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*

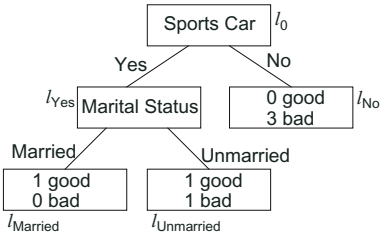

Fig. 1. A 3-anonymous decision tree

3 Inducing *k*-Anonymous Decision Trees

This section presents an algorithm which induces k-anonymous decision trees. The algorithm is based on the well-known ID3 algorithm [16] and on its extension, C4.5 [17]. ID3 applies greedy hill-climbing to construct a decision tree. Starting from a root that holds the entire learning set, it chooses the attribute that maximizes the information gain, and splits the current node into several new nodes. The learning set is then divided among the new nodes according to the value each tuple takes on the chosen attribute, and the algorithm is applied recursively on the new nodes.

The k-anonymity preserving equivalent of ID3, kADET (Algorithm 3.1), uses the same hill-climbing approach, with two changes: First, when considering

all possible splits of a node, kADET eliminates splits which would lead to k-anonymity breach. Second, the algorithm is not recursive. Instead, all the potential splits are considered in a single priority queue and the best one of all those that retain k-anonymity is picked. This method is required since k-anonymity is defined in terms of spans, which may include bins from several decision tree nodes. A decision regarding one node may thus influence other nodes.

3.1 *k*ADET Algorithm

The input of kADET is a private database T, the public attributes P, the private attributes Q, the class attribute C, and the anonymity parameter k. First, kADET computes the initial set of equivalence classes (bins) and spans: a single span containing all of the bins and all of the tuples if the class is private, and as many spans as bins, with each span containing a single bin and its tuple population if the class is public. If one of the spans contains less than k tuples from T, kADET returns *nil* and terminates. Otherwise, kADET creates the initial queue of possible splits, where each candidate split contains the root node and an attribute from P or Q. The queue is ordered according to the gain from each split. kADET then enters its main loop.

The main loop of kADET has the following steps: First, the most gainful candidate split $(node, attribute, gain)$ is popped out of the queue. If the node regarded in this candidate is already split, the candidate is purged. Otherwise, kADET tests whether splitting the node according to the suggested attribute would breach k-anonymity. If it would, then, again, this candidate is purged. However, if the attribute can be generalized, then a new candidate is inserted to the queue, this time with the generalized attribute. Finally, if k-anonymity is not breached, the node is split.

Several actions are taken in the splitting of a node: First, every bin of the parent node is split between the descendant nodes, according to the value of the splitting attribute. Accordingly, every span that contains this bin is updated with the new list of bins. The descendant nodes inherit the list of spans from their parent, and are added to the lists of nodes of those spans. If the splitting attribute is private, no further action is required, as the attacker cannot distinguish between the new bins. However, if the splitting attribute is public, then the attacker can use the split to distinguish tuples. Specifically, tuples that are routed to one of the new nodes will not be routed to its sibling nodes. Therefore, each of the spans in the split node, is split into new spans, one for each new descendant node. Each new span contains the bins from the original span, except for those of the sibling nodes. Likewise, the population of the original span is divided according to the value the tuples take on the splitting attribute. Nodes whose bins are contained in the new spans, and which are not descendant of the original node, are associated with all of the new spans.

Figure 2 demonstrates an execution of *kADET*, using the data in Table 1 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. 1.

Algorithm 1. k-Anonymous Decision Tree (kADET)

Input: T – private dataset, P – public attributes, Q – private attributes, C – the class attribute, k – anonymity parameter
procedure MAIN
  Create $root$ node in $Tree$
  Create in $root$ one bin b_c for each value $c \in C$ and divide T among the bins
  if $C \in P$ **then**
    Create one span S_c for every value $c \in C$,
      $S_c.Bins \leftarrow \{b_c\}$, $S_c.Population \leftarrow b_c.Population$, $S_c.Nodes \leftarrow \{root\}$
    set $root.Spans$ to the list of all spans
  else
    Create a single span s. Set $s.Bins$ to the list of all bins,
      $s.Population \leftarrow T$, $s.Nodes \leftarrow \{root\}$, $root.Spans \leftarrow \{s\}$
  if $\exists span \in root.Spans$ such that $0 < |span.Population| < k$ **then return** nil
  for $att \in P \bigcup Q \setminus \{C\}$ **do** add $(root, att, \text{gain}(root, att))$ to $Queue$
  while $Queue$ has elements with positive gain **do**
    Let $(n, a, gain) = \arg\max_{gain} \{Queue\}$
    if $n.sons \neq \emptyset$ **then continue**
    if $Breach\,(n, a, k)$ **then**
      if a has generalization a' **then** insert $(n, a', \text{gain}(n, a'))$ to $Queue$
    else Split(n, a)
  Set the Class variable in each leaf to the value with the largest bin.
  return $Tree$

---

procedure BREACH($node$, att, k)
  if $att \in Q$ **then return** $false$
  for $v \in att.values$ and $span \in node.Spans$ **do**
    if $0 < |\{t \in span.Population : t[att] = v\}| < k$ **then return** $true$
  return $false$

---

procedure SPLIT($node$, att)
  for $v \in att.values$ **do**
    Let $node.sons[v]$ be a new descendant node
    Let $node.sons[v].Bins[b]$ be a new bin, which refines $node.Bins[b]$ such that
      $node.sons[v].Bins[b].tuples \leftarrow \{t \in node.Bins[b].tuples : t[att] = v\}$
    Let $node.sons[v].Spans \leftarrow node.Spans$
  for $span \in node.Spans$ **do** replace each bin of the original node with its refinements, computed above, and add the new nodes to $span.Nodes$
  if $att \in P$ **then**
    for $span \in node.Spans$ **do**
      Remove $span$ from every node $n \in span.Nodes$
      for $v \in att.values$ **do**
        Create a new span s_v
        $s_v.Nodes \leftarrow span.Nodes \setminus \{node.sons[u] : u \neq v\}$
        $s_v.Bins \leftarrow span.Bins \setminus \{bin \in node.sons[u].Bins : u \neq v\}$
        $s_v.Population \leftarrow \{t \in span.Population : t[att] = v\}$
        Add s to $node.sons[v].spans$
        Add s to every node $n \in span.Nodes \setminus node.sons$

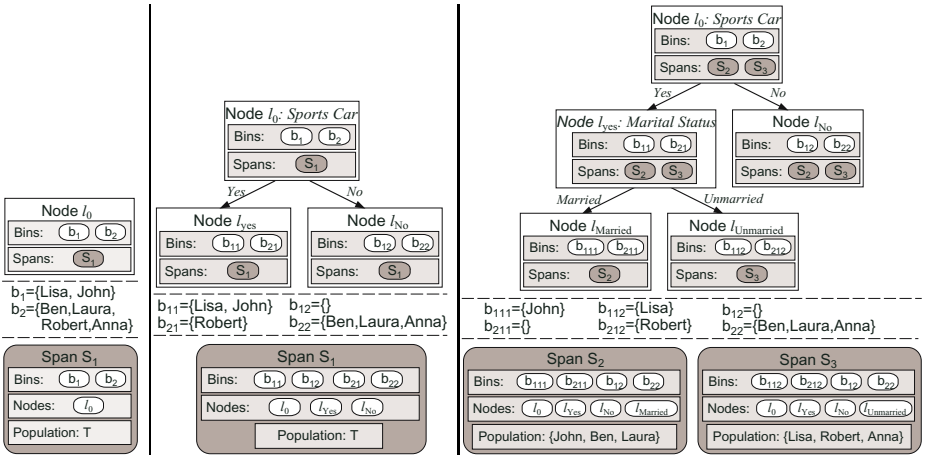

Fig. 2. Execution of $kADET$

3.2 Correctness and Overhead Analysis

The key to proving correctness of the algorithm is in showing that the population of each span, as computed by the algorithm, is the same as the one defined in Sect. 2: the set of tuples which, without knowledge of private attribute values, can be routed to the same set of bins. The proof is omitted due to lack of space.

The computational overhead incurred by the algorithm, respective to that of ID3, stems from the need to compute and track all of the different spans. In the worst case, the number of spans may reach the size of the public domain A. To see how, consider a tree in which all of the top nodes split on private attributes, until the number of leaves is equal to the number of public attributes and then, every leaf is split according to a *different* public attribute. The number of spans in the tree is equal to the size of A.

While this overhead is indeed high, it is inherent to the problem, because of the need to validate that every span is populated by k tuples or more. In practice, the number of spans will be much smaller. For example, when only the class attribute is private, the number of spans is the number of leaves.

3.3 From ID3 to C4.5

C4.5 was introduced by Quinlan [17] in order to extend and improve ID3. It implements better attribute scoring metrics (gain ratio instead of gain), error-based pruning, continuous attribute quantization, and treatment of missing values. All these extensions, other than the change of the scoring function – which has no effect on privacy – require careful analysis when used to extend kADET.

Pruning. C4.5 uses error-based pruning in two ways: discarding subtrees and replacing them with leaves, and replacing subtrees with one of their branches.

Using the first method is safe – undoing a split unifies equivalence classes, and may unify spans, meaning that the population of a span can only increase. The second method, however, may cause a k-anonymous tree to become non-k-anonymous, as it induces different spans with different populations. Therefore we avoid this technique in our implementation.

Continuous attributes. In the C4.5 algorithm, continuous attributes are handled by creating binary splits. The algorithm considers all the possible split points, and chooses the one with the best information gain. We implemented the same approach, adding the constraint that a split point should not cause a breach of k-anonymity.

Missing values. Missing values extend the k-anonymity model in ways which have not been modelled yet. It is not clear, for instance, whether a value that is missing in the learning examples would be missing in the data available to the attacker. We leave the extension of the k-anonymity model to missing values for further research.

4 Evaluation

To conduct our experiments we implemented the algorithms using the Weka package [18]. We use as a benchmark the Adults database from the UC Irvine Machine Learning Repository [19], 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. The class attribute is income level, with two possible values, $\leq$50K or >50K. 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 >50K). For the categorial attributes we use the same generalization hierarchies described in [10]. For the ID3 experiments we dropped the continuous attributes, because of ID3 limitations. The experiment was performed on a 3.0GHz Pentium IV processor with 512MB memory.

The anonymized decision trees algorithms use 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 4 seconds for ID3, and less than 10 seconds for C4.5.

4.1 Accuracy vs. Anonymity Tradeoffs

Our first goal is to assess the tradeoff between classification accuracy and the privacy constraint.

Figure 3 shows the classification error of the anonymous ID3 for various k parameters, compared to the classification error for ID3 and C4.5. 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 $k = 1000$ this attribute is discarded because of an anonymity breach, and the *Marital Status* attribute is chosen instead, yielding better classification.

We compared our results with those obtained using the top-down specialization (TDS) algorithm presented in [10], the goal of which is to produce

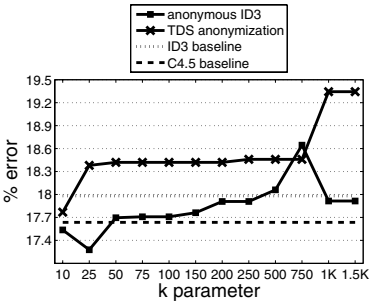

Fig. 3. Classification error vs. k parameter for ID3

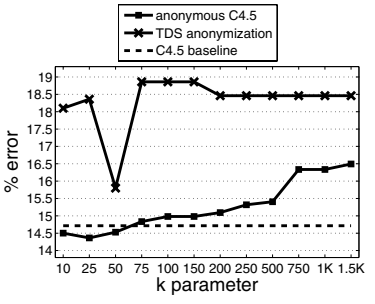

Fig. 4. Classification error vs. k parameter for C4.5

anonymized data useful for classification problems. The algorithm starts with the topmost generalization level, and iteratively chooses attributes to specialize, using a metric that measures the information gain for each unit of anonymity loss. The same generalization scheme is applied on all the tuples. We note that TDS uses both training and test data to choose a generalization. This may provide different generalization results, though not necessarily better or worse than those obtained when generalizing the training data alone. TDS results also appear in Fig. 3. 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.

Figure 4 shows a similar comparison using all 14 attributes of the Adult dataset with the anonymous C4.5 algorithm. The large size of the quasi-identifier affects the accuracy of the TDS generalization, and our algorithm reduces the classification error by an average of 3% with respect to TDS.

4.2 Privacy Risks and ℓ-Diversity

k-anonymity makes no restriction regarding private attribute values. Therefore, it is possible that a k-anonymous model would allow a complete inference of these values. In this section, our goal is to assess how many individuals are prone to immediate inference attacks and show how such attacks can be thwarted.

We look at the number of individuals for whom an attacker may infer the class attribute value with full certainty. This is the number of tuples associated with spans for which all the tuples share the same class. Because of space limits, we provide only the main figures. For the anonymous ID3, this number drops to zero only for values of k beyond 750, and even then the attacker may still be able to infer attribute values with high probability. The inference problem is less acute in the case of the anonymous C4.5, because of pruning. The number of exposed tuples drops to zero at $k = 75$, and is very low (below 0.5%) even for smaller values of k.

The ℓ-diversity model [20] suggests solving the inference problem by requiring a certain level 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 algorithms by replacing the $Breach()$ function with one that checks the entropy ℓ-diversity constraint, ruling out splits that violate this constraint. Note that the parameters for k-anonymity and ℓ-diversity are not comparable. In particular, as there are only two class values, the best we can hope for is entropy 2-diversity. This is achieved when there is equal chance for each class value. However, for $\ell < 2$, entropy ℓ-diversity limits the attacker's confidence in inference attacks. The *confidence limit* is the maximal probability of any private value for any individual. The data owner can control the confidence limit by manipulating the ℓ parameter. For example, to deny the attacker the ability to infer a class value with confidence greater than 85%, entropy higher than $0.85 \times \log(1/0.85) + 0.15 \times \log(1/0.15) = 0.61$ should be maintained. This amounts to applying entropy ℓ-diversity with $\ell = 1.526$ ($\log 1.526 = 0.61$).

Following this discussion, Figures 5 and 6 display the tradeoff between the confidence limit and the accuracy of the induced decision trees. So long as the confidence threshold is high enough, it is possible to induce 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. In the case of ID3, every split of the root node results in a node with confidence greater than 85%. Therefore, a confidence limit of 85% or lower prohibits the induction of a useful decision tree. The additional attributes available to the C4.5 algorithm allow the boundary to be stretched to a lower confidence threshold.

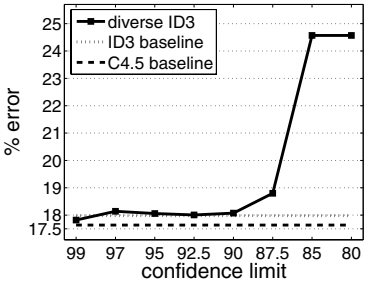

Fig. 5. Confidence level vs. error rate for ID3, 9 attributes

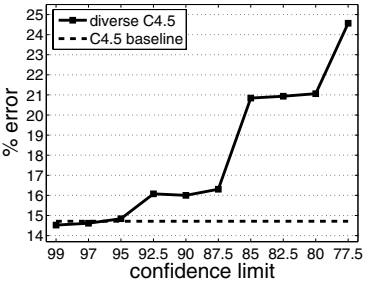

Fig. 6. Confidence level vs. error rate for C4.5, 15 attributes

5 Conclusions

In this paper we presented decision tree induction algorithms which guarantee k-anonymous output. Using our definitions, it is possible to introduce similar constraints in other data mining algorithms as well. Another interesting use of this method is promoted by the ability to construct a table that is equivalent to a data mining model. Such a table would maintain k-anonymity, while preserving

the data patterns evident in the data mining model. Hence data mining algorithms can be used as templates for pattern-preserving anonymization schemes.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proc. of the ACM SIGMOD Conference on Management of Data, ACM Press (2000) 439–450
2. Du, W., Zhan, Z.: Building decision tree classifier on private data. In: Proc. of CRPITS'14, Darlinghurst, Australia, Australian Computer Society, Inc. (2002) 1–8
3. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Proc. of CRYPTO '00, London, UK, Springer-Verlag (2000) 36–54
4. Vaidya, J., Clifton, C.: Privacy-preserving decision trees over vertically partitioned data. In: DBSec. (2005) 139–152
5. Kantarcioğlu, M., Jin, J., Clifton, C.: When do data mining results violate privacy? In: Proc. of ACM SIGKDD, NY, USA, ACM Press (2004) 599–604
6. US Dept. of HHS: Standards for privacy of individually identifiable health information; final rule (2002)
7. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proc. of PODS '04, New York, NY, USA, ACM Press (2004) 223–228
8. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Approximation algorithms for k-anonymity. In: Journal of Privacy Technology (JOPT). (2005)
9. Bayardo Jr., R.J., Agrawal, R.: Data privacy through optimal k-anonymization. In: Proc. of ICDE. (2005) 217–228
10. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: Proc. of ICDE. (2005)
11. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: Proc. of ACM SIGKDD. (2002) 279–288
12. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proc. of ICDE. (2006)
13. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems **10**(5) (2002) 571–588
14. Atzori, M., Bonchi, F., Giannotti, F., Pedreschi, D.: k-anonymous patterns. In: Proc. of PKDD. (2005) 10–21
15. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proc. of SIGMOD, NY, USA, ACM Press (2005) 49–60
16. Quinlan, J.R.: Induction of decision trees. Machine Learning **1**(1) (1986) 81–106
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd edn. Morgan Kaufmann, San Francisco (2005)
19. D.J. Newman, S. Hettich, C.B., Merz, C.: UCI repository of machine learning databases (1998)
20. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. In: Proc. of ICDE. (2006)