TRANSACTIONS ON DATA PRIVACY 5 (2012) 433–467

A Knowledge Model Sharing Based Approach to Privacy-Preserving Data Mining

Hongwei Tian, Weining Zhang, Shouhuai Xu and Patrick Sharkey
Department of Computer Science, University of Texas at San Antonio
{htian, wzhang, shxu, psharkey}@cs.utsa.edu

Abstract. Privacy-preserving data mining (PPDM) is an important problem and is currently studied in three approaches: the cryptographic approach, the data publishing, and the model publishing. However, each of these approaches has some problems. The cryptographic approach does not protect privacy of learned knowledge models and may have performance and scalability issues. The data publishing, although is popular, may suffer from too much utility loss for certain types of data mining applications. The model publishing is lacking of efficient algorithms for practical use in a multiple data source environment.

In this paper, we present a knowledge model sharing based approach which learns a global knowledge model from pseudo-data generated according to anonymized knowledge models published by local data sources. Specifically, for the anonymization of knowledge models, we present two privacy measures for decision trees and an algorithm that obtains an anonymized decision tree by tree pruning. For the pseudo-data generation, we present an algorithm that generates useful pseudo-data from decision trees. We empirically study our method by comparing it with several PPDM methods that utilize existing techniques, including three methods that publish anonymized-data, one method that learns anonymized decision trees directly from the original-data, and one method that uses ensemble classification. Our results show that in both single data source and multiple data source environments and for several different datasets, predictive models, and utility measures, our method can obtain significantly better predictive models (especially, decision trees) than the other methods.

Keywords. privacy-preserving data mining, knowledge model sharing, pseudo-data generation, ℓ-diversity, decision tree pruning

1 Introduction

A huge amount of data has been collected by organizations of all kinds. Many applications in areas such as healthcare, medical science, financial services, e-commerce, and national security can benefit from useful patterns in the data. Data mining techniques provide powerful tools to extract such patterns from the data. However, since the data can contain sensitive personal information, organizations are prohibited by law from releasing the raw

*Some preliminary result of this paper was published at a workshop [1].
†This work was supported in part by NSF research grant IIS-0524612.

data. Several approaches have been developed to perform the so called privacy-preserving data mining (PPDM).

One approach is to have data sources execute specially designed cryptographic data mining algorithms [2, 3]. This approach guarantees the privacy during the mining process and produces accurate global knowledge models. However, it does not concern with privacy disclosure caused by the learned knowledge models [4]. In addition, it requires new algorithms for each data mining task for which conventional algorithms are already available. It also has certain performance and scalability issues.

Another approach is to publish anonymized-data [5, 6, 7, 8, 9, 10, 11, 12]. In this approach, privacy is protected by altering the original-data. For example, a perturbation [5] can change the original-data by introducing random noise or by random data replacement, and a generalization [10] can change the original-data by replacing specific values with semantically more general values. The anonymization is based on some privacy measures, such as k-anonymity [10] or ℓ-diversity [11]. Due to the intuitiveness of these privacy measures, data publishing has been extensively studied as practical solutions for PPDM and resulted in many efficient and scalable methods. Since the anonymized-data can be used by conventional data mining algorithms to learn various kind of knowledge models, this approach is flexible and general. However, the anonymized-data may have low utility for certain types of data mining applications.

A third approach is to publish some models of the original-data, such as (some statistics of) a set of clusters [13] or contingency tables [14], that satisfy a given privacy requirement, such as k-anonymity or differential privacy [15]. The motivation is twofold. On the one hand, since these models are more abstract than data, they can help to further reduce the risk of privacy disclosure. On the other hand, since these models fit well with online analytic processing (OLAP) and aggregate queries, they result in better utility for these applications. Although promising, this approach currently lacks efficient algorithms for practical use.

A more practical alternative is to publish anonymized knowledge models, such as k-anonymity decision trees [16]. It has been shown in [16] that an anonymized decision tree may be obtained more efficiently than anonymized-data and that the published decision trees can have a better quality than those learned from the anonymized-data. However, there are two drawbacks of this method. First, to learn an anonymized knowledge model directly from the original-data requires the re-design of data mining algorithms and it is difficult to balance privacy and utility within these algorithms. Second, this method works only for a single data source. Although an ensemble method, such as bagging [17], may be used to obtain a global classifier by aggregating decision trees published by multiple data sources, it is not yet clear how such an aggregation can be applied to other types of knowledge models.

In this paper, we propose a *knowledge model sharing* based approach to PPDM, in which a global knowledge model is learned from anonymized knowledge models published by local data sources. To obtain anonymized knowledge models, data sources learn knowledge models using conventional algorithms and modify the learned model to satisfy certain anonymity requirements. To obtain global knowledge models, the published local knowledge models are used to generate pseudo-data and conventional data mining algorithms are then applied on the pseudo-data.

It is worth mentioning that although it is possible for a local data source to publish the pseudo-data generated using our method, we do not recommend it over the publishing of the local knowledge model for two reasons. First, the local knowledge model is anonymized before publishing to protect privacy, but due to the nature of the pseudo-data generation,

it is not guaranteed that the pseudo-data generated from the anonymized local knowledge model would also satisfy the same anonymity requirement. Thus it may not be safe to publish the pseudo-data. However, it causes no privacy concern if the pseudo-data is generated by the data miner or the adversary. Secondly, publishing a local knowledge model may provide a better service to users who are interested in the knowledge model learned from a single data source, because the same knowledge model may not always be obtained from the published pseudo-data.

1.1 Our Contributions

Specifically, we make the following contributions.

1. We present a framework in which each data source publishes an anonymized knowledge model, and a closely related global knowledge model is learned from the pseudo-data generated from the published local knowledge models. The two challenging issues of this framework are the privacy of knowledge models and the generation of high-quality pseudo-data.

2. For the privacy of knowledge models, we propose two privacy measures for decision trees assuming the class labels represent sensitive information. Both measures, the k-anonymity and the ℓ-diversity, are adapted from data publishing. To the best of our knowledge, this is the first attempt to measure the privacy of a knowledge model.

3. We present an algorithm that obtains an anonymized decision tree by pruning the tree to satisfy a given anonymity requirement. This technique can be viewed as a special type of generalization that preserves important patterns in the original-data. Compared to the method that learns anonymized decision trees directly from the original-data, such as [16], our method can preserve more crucial paths of the decision tree, therefore results in better utility.

4. For pseudo-data generation, we present an efficient heuristic algorithm which uses the paths of the decision tree as templates to generate pseudo tuples. This method can be easily extended to generate pseudo-data from disjoint classification rules. The pseudo-data produced by this algorithm is better than those generated from random sampling or contingency tables because the paths in decision trees emphasize the most important patterns in the original-data that are critical for learning a good-quality global knowledge model.

5. We empirically study our method by comparing it with several PPDM methods that utilize existing techniques. These include three methods that publish anonymized-data, one method that learns anonymized decision trees directly from the original-data, and one method that uses ensemble classification. For tree pruning, we compare predictive models learned from pseudo-data, which is generated from the pruned tree, with those learned from the original-data or from the anonymized-data. We also compare the pseudo-data directly with the anonymized-data. For pseudo-data generation, we compare predictive models learned from the pseudo-data with those learned directly from the original-data, and also with those obtained by the ensemble method. In addition to decision trees, we also consider Naive Bayes classifiers and conjunctive rules, as global predictive models. Our results show that in both single data source and multiple data source environments and for several different datasets,

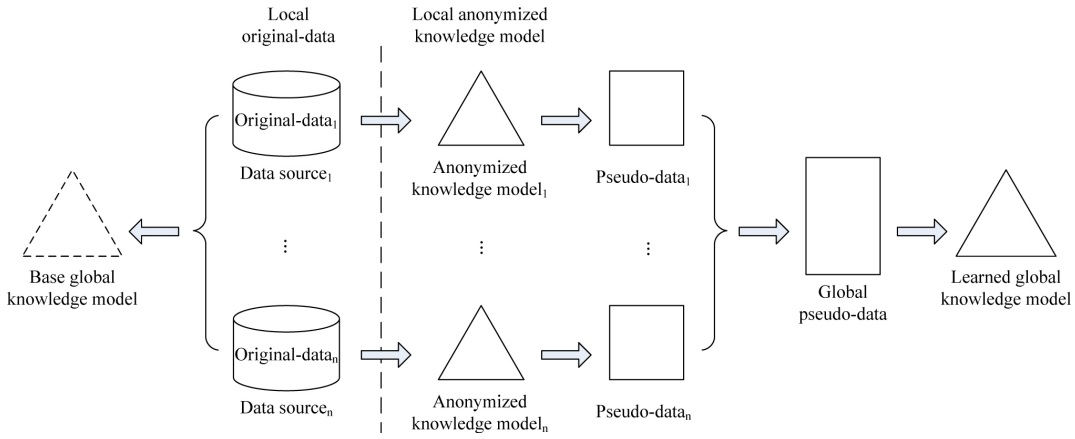


Figure 1: A framework of knowledge model sharing approach to PPDM. Individual data source publishes an anonymized knowledge model learned from its local original-data. The data miner generates pseudo-data from published knowledge models and learn a global knowledge model from the pseudo-data. The goal is for the global knowledge model to have a quality as high as the base global knowledge model.

predictive models, and utility measures, our method can obtain significantly better predictive models (especially for decision trees) than the other methods.

1.2 Roadmap

The rest of this paper is organized as follows. In Section 2, we present a framework of the new PPDM approach. In Section 3, we present privacy measures for decision tree. In Section 4, we present an algorithm that prunes a decision tree to satisfy a given anonymity requirement. In Section 5, we define the problem of pseudo-data generation from a decision tree and present a path-based pseudo-data generation algorithm. In Section 6, we review several techniques that may provide alternative implementation to several parts of the framework. In Section 7, we present results of our empirical study. In Section 8, we discuss related work. We draw conclusions and present future work in Section 9.

2 A Framework of Knowledge Model Sharing

Figure 1 shows a framework of the knowledge model sharing based approach of PPDM. It involves several types of data and knowledge-models and has two major components.

2.1 Types of Data and Knowledge-Models

There are following types of data and knowledge models at the local and the global levels of this framework.

1. Local data. A data source may have three types of local data, namely, *local original-data*, which is the private data at the data source, *local anonymized-data*, which is the anonymized-data published by a data source, and *local pseudo-data*, which is a pseudo-data generated from a knowledge model published by the data source.

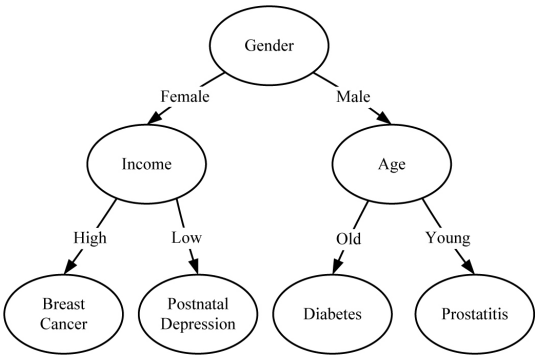


Figure 2: Example decision tree that discloses privacy

2. Local knowledge models. There are two types of local knowledge models, both are learned from the local original-data. A *local original knowledge model* is learned from the local original-data using a conventional data mining algorithm. An *local anonymized knowledge model* is either learned from the local original-data using a special privacy-preserving data mining algorithm or obtained from an local original knowledge model by a privacy-preserving alteration method.

3. Global data. A global data is an integration (or aggregation) of local data. There are two types of global data, namely, the *global original-data*, which is the integration of local original-data, and the *global pseudo-data*, which is the integration of local pseudo-data. In the knowledge model sharing framework, the global original-data does not exist physically since data sources are not allowed to disclose their local original-data. However, it exists conceptually and can be used to define and to measure the utility of PPDM methods. The global pseudo-data is used by the data miner to perform data mining.

4. Global knowledge models. A global knowledge model is learned from a global data. The *base global knowledge model* is learned from the global original-data using a conventional data mining algorithm. Again, base global knowledge models do not physically exist in the framework, but is important and useful in defining and measuring the quality of data mining results. The *learned global knowledge model* is obtained from the global pseudo-data and is the output of the framework.

2.2 Components of the Framework

The two components of the framework deal respectively with the publishing of local knowledge models and the mining of pseudo-data.

2.2.1 Publishing Local Knowledge Models

In this component, each data source publishes a local knowledge model of the same or a similar type as required by a given data mining task. The published knowledge model must not only capture the characteristic of the local original-data but also protect privacy. This is challenging because knowledge models learned from the local original-data may disclose private information.

Example 1. Suppose the decision tree in Figure 2 is published and all training tuples in a leaf node belong to the same class. If the Disease is a sensitive information and a target person is known to be an old man, the adversary can use the decision tree to classify the target person and know with a high certainty that the target person has Diabetes. On the other hand, if the Income is sensitive and a target person is known to be a woman having a Breast Cancer, the adversary can match the information to the leftmost path of the tree and know with a high probability that this person has a high income.

To effectively implement this component, we must define appropriate privacy measures for various types of knowledge models and develop effective methods to obtain knowledge models that satisfy given privacy requirements.

2.2.2 Mining Pseudo-Data

This component allows a data miner to generate a set of pseudo-data, one from each published privacy-preserving knowledge model, combine these local pseudo-data into a global pseudo-data, and learn a global knowledge model using conventional data mining algorithms. The goal is to allow the data miner to learn any type of knowledge model from the global pseudo-data.

A challenge is to generate high-quality pseudo-data for different data mining tasks. To effectively implement this component, we need to define task-specific utility measures for pseudo-data and design effective and efficient algorithms to generate pseudo-data that minimizes the utility loss.

3 Privacy Measure for Decision Trees

We consider decision trees in which the class attribute[1] contains sensitive information. One way to measure the privacy of decision trees is to adapt the k-anonymity and ℓ-diversity measures of data tables for decision trees.

3.1 k-anonymity and ℓ-diversity for Data

For tabular data publishing, each tuple represents a person and the attributes of a table are partitioned into three types: *personal identity (PI)*, *quasi-identifier (QI)*, and *sensitive attribute (SA)*, where PI uniquely identifies individuals, SA contains the private information, and QI does not reveal personal identity by itself but can reveal the identity of some persons if combined with some publicly available data. When data is published, it is not sufficient to just remove PI, because of the *linking attack* whereby the adversary uses the QI and the public data to link a target person to a specific SA value.

A common technique to prevent a linking attack is the generalization [11, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27], which alters QI values according to some taxonomy or generalization hierarchy. The basic idea is to replace (recode or generalize) specific QI values by more general values in the given taxonomies. As a result, tuples that are distinct in the private table become indistinguishable in the anonymized table and the anonymized table can be partitioned into *equivalence classes (ECs)*, where each EC contains tuples that have the same QI value. A privacy measure of the anonymized table is the notion of k-anonymity [20, 22, 24].

[1] We assume there is only one sensitive attribute.

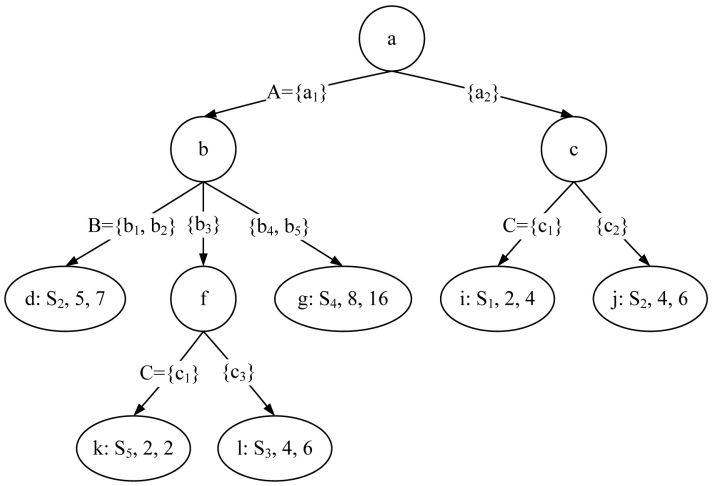


Figure 3: A sample decision tree with leaf labeled by (class label, hit count, miss count)

Definition 2. (k**-**anonymity [20]) A table satisfies k-anonymity if every EC in the table contains at least k tuples.

The k-anonymity prevents the linking attack by making each individual indistinguishable from at least $k-1$ other individuals in the anonymized table. Nevertheless, it does not prevent *homogeneity attack* whereby the adversary applies some background knowledge about SA values to identify SA value of a target person. For example, if the EC of a target person contains only one SA value, the adversary can infer the SA value of the target person even if the EC contains k or more tuples. To prevent homogeneity attack, the notion of ℓ-diversity was proposed in [11], which requires SA values in each EC to be well-represented. Widely accepted measures of the well-representedness include the recursive (c, ℓ)-diversity [11] and simple ℓ-diversity [18, 23].

Definition 3. (ℓ-diversity [11, 18]) A table satisfies the *recursive* (c, ℓ)*-diversity* (resp. *simple* ℓ*-diversity*) if for each equivalence class E, $f_1 < c \times \sum_{i=\ell}^{m} f_i$ (resp. $f_1 \leq 1/\ell$), where f_i is the frequency of the i^{th} most frequent SA value in E, m is the number of distinct SA values in E, and $c \geq 1$ is an integer constant.

Intuitively, the ℓ-diversity requires that there are at least ℓ distinct SA values in an EC. It also limits the frequency of the most frequent SA value in the EC, therefore, the frequency of any other SA value is also limited. Thus, it prevents the distribution of SA values from being very skewed. For recursive (c, ℓ)-diversity, it restricts the difference between f_1 and the sum of the frequency of SA values except the first $\ell-1$ most frequent ones. It is not good if f_1 is too large or $\sum_{i=\ell}^{m} f_i$ is too small. For simple ℓ-diversity, it restricts that f_1 should not be greater than an absolute constant $1/\ell$.

3.2 Information Contained in a Decision Tree

A decision tree consists of labeled nodes and edges (see an example in Figure 3). As a predictive mode, it can classify unseen tuples by matching the attribute values of a tuple to the edge labels of a path in the decision tree, starting at the root. The class label in the

leaf node at the end of the path predicts the class of the tuple. A decision tree is also a descriptive model that describes properties of the training data tuples. This provides the basis for generating pseudo-data that preserve important properties of the local original-data.

In the following, we formalize the information typically found in a decision tree, which significantly affects the privacy.

Definition 4. (Edge Label) An edge label is a predicate of the form $A = a$, where A is an attribute and a is either a set of values (if A is categorical) or a set of intervals (if A is numeric)[2]. The labels of outgoing edges of a node have the same attribute and their values form a partition of the domain of the attribute.

Definition 5. (Node Information) The label of a node is of the form $(class, hit, miss)$, where $class$ is the label of the majority (or plurality) class among training tuples in the node; hit is the number[3] of training tuples of the majority class; and $miss$ is the number of misclassified training tuples (of some non-majority class).

Example 6. In Figure 3, the label of node d indicates that there are 12 tuple in d, 5 of them belong to the major class S_2, the remaining 7 are misclassified.

Definition 7. (Path Information) The label of a root-to-leaf path is a set which for each attribute A appearing in the path contains a unique edge label $A = a$ where a is the set of the most specific values of A in the entire path. The information of a path includes the node information of its leaf and the information in the path label.

Notice that although a categorical attribute can appear in at most one edge label in a path, a numeric attribute may appear in multiple edge labels. However, in this case, the values in these edge labels will be nested intervals. The smallest among these intervals is the one included in the path label.

In the remaining of this paper, we use $v.a$ to denote information a (such as class, hit, or miss) of a node v, $label(p)$ the label of path p, $attr(p)$ the set of attributes in path label of p, and $value(A, p)$ the set of values of an attribute $A \in attr(p)$.

Example 8. In the leftmost path p in Figure 3, the leaf node is d, thus information of p contains $p.class = d.class = S_2$, $p.hit = d.hit = 5$, and $p.miss = d.miss = 7$. It also has the label $label(p) = (A = \{a_1\} \wedge B = \{b1, b2\})$. Thus $attr(p) = \{A, B\}$, $value(A, p) = \{a_1\}$ and $value(B, p) = \{b_1, b_2\}$.

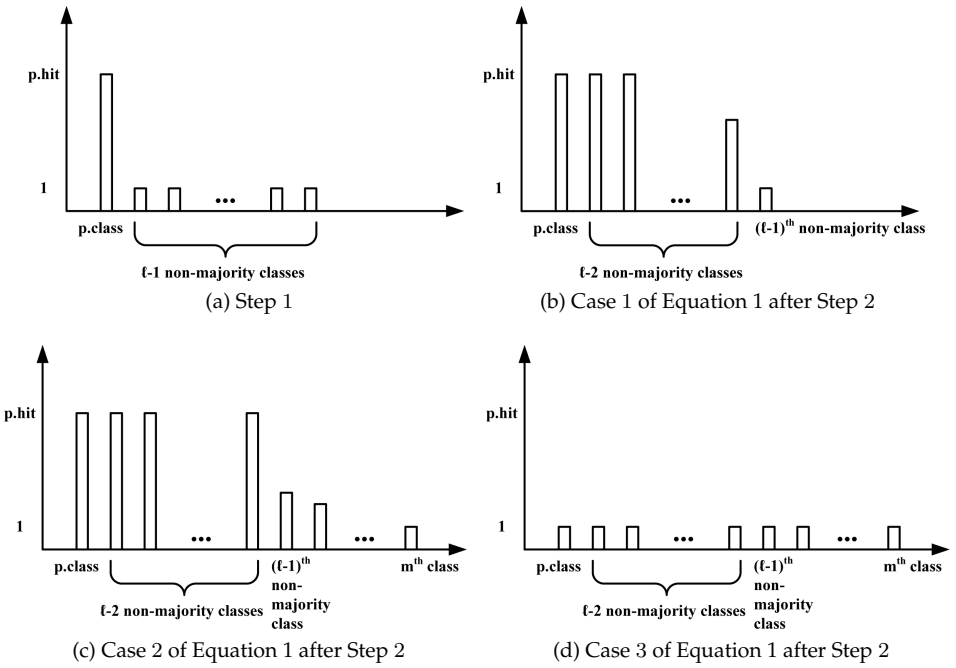
To predict the class of an unseen tuple t, we simply match the attributes of t to the edge labels of paths. If there is a path p such that $t[A] \in value(A, p)$ for every attribute A in $attr(p)$, t belongs to class $p.class$.

3.3 k-anonymity and ℓ-diversity for Decision Trees

Paths of a decision tree form a partition of the training tuples in the same way that QI equivalence classes form a partition of an anonymized table. Indeed, decision tree learning algorithms partition training tuples at each tree node according to values in a split attribute.

[2]This decision tree may have been pruned to preserve privacy, causing edge labels to contain a set of values (or intervals).

[3]To ease the presentation, we consider only the actual counts. It is straightforward to extend this to relative frequencies.



(a) Step 1

(b) Case 1 of Equation 1 after Step 2

(c) Case 2 of Equation 1 after Step 2

(d) Case 3 of Equation 1 after Step 2

Figure 4: Examples of constructing class distribution for Equation 1. The X-axis is a list of classes. Except $p.class$, other classes are randomly ordered. The Y-axis is the number of tuples in a class. The $p.miss$ misclassified tuples will be distributed to at least $\ell - 1$ classes.

Therefore, tuples in a leaf node will have the same values in (the split) attributes encountered along the path. Based on this observation, we adapt privacy measures of data tables to measure the privacy for decision trees.

Definition 9. (decision tree k-anonymity) A decision tree satisfies the k-anonymity if $p.hit+ p.miss \geq k$ for each path p of the decision tree.

Example 10. The decision tree in Figure 5 satisfies 3-anonymity.

Definition 11. (decision tree (c, ℓ)-diversity) A decision tree satisfies (c, ℓ)-diversity if every path p satisfies $p.miss \geq \ell - 1$, $p.hit < c \times p.tail$ and

$$p.tail = \begin{cases} 1, & \text{if } 0 \leq q \leq \ell - 2 \text{ and } p.hit > 1; \\ p.miss - (\ell - 2) \times p.hit, & \text{if } q > \ell - 2 \text{ and } p.hit > 1; \\ p.miss - (\ell - 2), & \text{if } p.miss \geq \ell - 1 \text{and } p.hit = 1 \end{cases} \quad (1)$$

where $q = [p.miss - (\ell - 1)]/(p.hit - 1)$.

Intuitively, at the leaf node of path p, $p.hit$ tuples belong to the majority class $p.class$ and $p.miss$ tuples belong to other classes (regarded as misclassified). Notice that $p.miss \geq \ell - 1$ is a necessary condition (not sufficient condition) for ℓ-diversity. It guarantees to have at least ℓ different classes. In Definition 11, the frequency of the most frequent class, $p.hit$, should also be restricted to avoid the skew distribution as required by Definition 3. Notice that the condition $p.hit < c \times p.tail$ is obtained directly from the condition of recursive

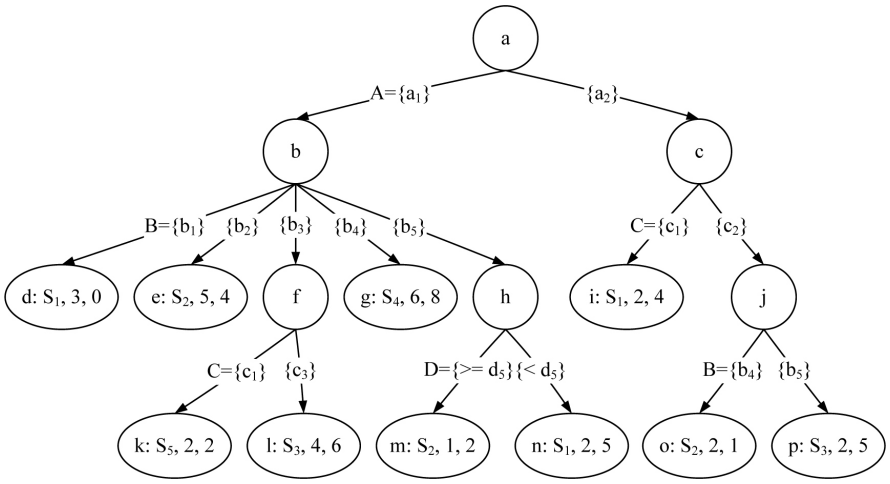


Figure 5: An example of published decision trees

(c, ℓ)-diversity in Definition 3 with $f_1 = p.hit/(p.hit+p.miss)$ and $\sum_{i=\ell}^{m} f_i = p.tail/(p.hit+p.miss)$.

The diversity of a path must be measured according to the class distribution of the $p.miss$ misclassified tuples. Since this information is not given in the leaf node of the path, Definition 11 assumes a highly skew distribution[4] that has

1. at least $\ell - 1$ non-empty non-majority classes; and

2. the maximum number of non-majority classes that contains $p.hit$ misclassified tuples.

For any path, this class distribution can be obtained as follows. First, randomly select $\ell - 1$ non-majority classes. Then, assign one misclassified tuple to each of these classes (so that they are not empty). This step guarantees that there are at least ℓ classes and the result is illustrated in Figure 4a. The number of remaining unassigned misclassified tuples is $p.miss - (\ell - 1)$, which is the numerator of q in Equation 1.

Next, for each non-majority class, assign to it another $p.hit - 1$ misclassified tuples as long as there are more left. If in the end all $\ell - 1$ non-majority classes have $p.hit$ tuples and there are still unassigned misclassified tuples, assign them evenly to non-majority classes that are still empty. The results after this step correspond to the three cases in Equation 1. Figure 4b shows the first case that there is no misclassified tuple left after they are assigned to the first $\ell - 2$ non-majority classes, thus the $p.tail$ only contains the number of tuples in the ℓ^{th} most frequent class, i.e., the $(\ell - 1)^{th}$ non-majority class. Obviously, this non-majority class has the only tuple assigned in the first step. Figure 4c shows the second case that all the first $\ell - 2$ non-majority classes have been assigned $p.hit$ tuples (1 tuple from Step 1 and $p.hit - 1$ tuples from Step 2), thus $p.tail$ includes the number of all the misclassified tuples except those assigned into the first $\ell - 2$ non-majority classes. Figure 4d shows a third and trivial case that each of the first $\ell - 1$ non-majority classes has $p.hit = 1$ tuple after the first step. This case is not included into the second case because of the constraint on q.

[4]We do not assume a uniform distribution because it makes (c, ℓ)-diversity much easy to satisfy, therefore, weakens the privacy.

Example 12. In Figure 5, the decision tree does not satisfy the $(3,3)$-diversity because none of the paths from the root to leaf nodes d, e, g, and o satisfies the requirement. Notice that paths from the root to leaf nodes k, l, m, n, i, and p have already satisfied the privacy requirement.

For instance, the path $A = a_1 \wedge B = b_b \wedge D < d_5$ satisfies $(3,3)$-diversity because its class distribution is $< 2,2,2,1 >$. In the procedure to obtain the class distribution, $\ell - 1 = 2$ non-majority classes are randomly selected and one tuple is assigned to each of the two non-majority classes. Since there are sufficient misclassified tuples left, $p.hit - 1 = 1$ tuple is assigned to each of the two non-majority classes. Since there is still one misclassified tuple left, it is assigned to a non-majority classes newly randomly selected. In this case, $p.tail$ is 3 (the second case in Equation 1). Also, the path $A = a_1 \wedge B = b_3 \wedge C = c_1$ satisfies the $(3,3)$-diversity since the class distribution is $< 2,1,1 >$, that is, two tuples in the majority class S_5 and one tuple in each of the two non-majority classes. In this case, $p.tail = 1$ (the first case in Equation 1).

The path $A = a_1 \wedge B = b_1$ does not satisfy the $(3,3)$-diversity because of $p.miss = 0 < \ell - 1 = 2$. The path $A = a_1 \wedge B = b_2$ also does not satisfy the $(3,3)$-diversity because its class distribution is $< 5,3,1 >$ and its tail is 1 (the first case in Equation 1), thus, $p.hit = 5 > c \times p.tail = 3$.

Definition 13. (Decision Tree simple ℓ-diversity) A decision tree satisfies the simple ℓ-diversity if every path p satisfies $p.miss \geq \ell - 1$ and $p.hit \leq (p.hit + p.miss)/\ell$.

Intuitively, Definition 13 is adapted from the simple ℓ-diversity in Definition 3, thus the frequency of the most frequent class $p.hit/(p.hit + p.miss)$ should not be greater than $1/\ell$.

Example 14. In the decision tree in Figure 5, the path $A = a_1 \wedge B = b_4$ satisfies simple 2-diversity, and the path $A = a_2 \wedge C = c_1$ satisfies simple 3-diversity.

In the following sections, we only use the ℓ-diversity measures for decision tree publishing.

4 Anonymize Decision Trees By Pruning

Given an anonymity requirement, a data source may produce an anonymized decision tree by learning an ordinary decision tree and pruning it to satisfy the anonymity requirement. In this section, we present a tree pruning algorithm.

If a path in the decision tree does not satisfy a given anonymity requirement, we can trim it to improve the privacy (due to the change of information in the resulting path). The pruning can also reduce the classification accuracy of the tree. Since it is very difficult to obtain an optimal tree pruning (with a minimum loss of classification accuracy), we present an anonymity-guided tree pruning (ATP) algorithm (see Figure 6) based on several heuristics.

One heuristic is to trim paths at nodes that are as close to leaf nodes as possible. The idea is to preserve the paths to the maximum degree. Using this heuristic, the ATP algorithm prunes the tree in a bottom-up level traversal (lines 4-13) starting at the level next to the bottom of the tree.

A simple way to prune a path is to remove the entire subtree at a chosen non-leaf node. This method may cause too much loss of utility. Therefore, ATP merges a few subtrees below that non-leaf node. Specifically, if a path does not satisfy the anonymity requirement, ATP will merge its leaf-node with some siblings of the leaf node (line 8). This merge will

Algorithm Anonymity-guided Tree Pruning (ATP)
Input: a decision tree T and an anonymity requirement α
Output: an altered T that satisfies α
Method:
1. for each node v in the depth-first traversal of T
2. if v is a leaf node then test v against α;
3. else compute $v.hit$, $v.miss$ and $v.class$;
4. go to the level next to the maximum level of T;
5. while T has more than two nodes and not all leaf nodes *pass* the test do
6. for each non-leaf node v at the current level do
7. while v has more than one child and some of them failed the test do
8. merge a failed child with some siblings;
9. test the resulted node against α;
10. if v has a single child then
11. replace v by the child;
12. else mark v *pass*;
13. move up into the next level;
14. if T has more than two nodes return T, else return an empty tree;

Figure 6: Anonymity-guided Tree Pruning Algorithm

replace the subtrees rooted at these nodes by a new leaf-node introduced as a new child of the non-leaf node. The node information of the new leaf node will be obtained from the subtrees it replaces. If the new leaf node is the only child of the non-leaf node, it will also replace the non-leaf node (line 11). The heuristics used by ATP for selecting siblings to merge are given in Section 4.2.

To determine the nodes to be merged, each node is tested against the anonymity requirement α: a leaf node is tested during a depth-first traversal (lines 1-3) and a non-leaf node is tested during the bottom-up level traversal. A leaf node passes the test if it satisfies α. A non-leaf node passes the test if all of its children pass the test. To prepare for possible merges, node information is calculated for non-leaf nodes by propagating information from leaf nodes in the depth-first traversal. The detail is given in Section 4.1.

Upon termination, the algorithm returns a decision tree satisfying the anonymity requirement. This tree either has two or more nodes or is empty (line 14).

The time complexity of this algorithm is $O(|V| + |E|)$, where V and E are the sets of nodes and edges in the tree T, respectively. In the following we discuss the details of node information propagation and the selection of nodes to merge.

4.1 Propagating Node Information

The node information of non-leaf nodes is calculated bottom-up in the depth-first traversal. Assume that a non-leaf node v has k children $\{u_1, \ldots, u_k\}$ and there are a total of m classes $\{c_1, \ldots, c_m\}$. To determine the majority class at v, we need to know the number of tuples in each class. If the training tuples in the subtree of v are available, we can simply count the tuples in each class. However, even if training tuples are not given, we can still estimate the number of tuples for each class.

Intuitively, if the majority class of child node u_i is c_j, this child node must contribute $u_i.hit$ tuples to class c_j in the parent v. If the majority class of u_i is not c_j, the number of

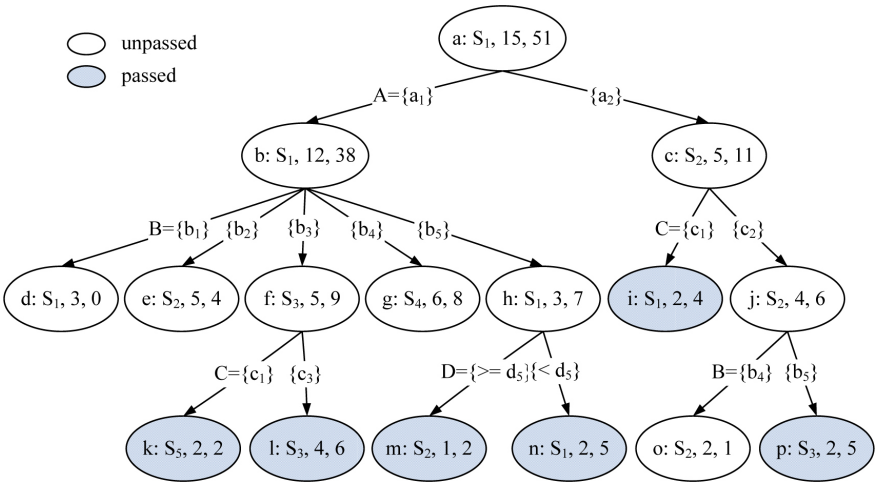


Figure 7: A snapshot of the decision tree in Figure 5 after the depth-first traversal of ATP with the privacy requirement of $(3, 3)$-diversity.

misclassified tuples in class c_j in u_i can be calculated based on a known class distribution or estimated from the uniform distribution. It is worth pointing out that although it is possible to keep track of class distribution in a decision tree, doing so would require redesign of conventional decision tree learning algorithms. On the other hand, to obtain exact class distribution of a conventional decision tree learned from the local data, one has to re-scan and partition the local data according to the decision tree. This will incur a heavy overhead. Thus, in this paper, the number of tuples in class c_j is estimated from the uniform distribution as follows.

$$hit_{c_j} = \sum_{u_i.class=c_j, 1\le i\le k} u_i.hit + \frac{1}{m-1} \cdot \sum_{u_i.class\neq c_j, 1\le i\le k} u_i.miss \tag{2}$$

where $u_i.miss/(m-1)$ is the expected number of misclassified tuples in u_i that belongs to class c_j. In this way, the estimated counts can propagate from child nodes to the parent node in the depth-first traversal.

Once the count of each class is obtained, the node information of v can be calculated as follows.

$$\begin{aligned} v.class &= c = \text{maxarg}_{c_j, 1\le j\le m}\{hit_{c_j}\} \\ v.hit &= \lceil hit_c \rceil \\ v.miss &= \sum_{1\le i\le k} (u_i.hit + u_i.miss) - v.hit \end{aligned} \tag{3}$$

Equations (2) and (3) can also be used to compute node information for nodes that result from merges. In this case, the new node can be viewed as the parent of those nodes being merged.

Example 15. In Figure 7, leaf nodes k, l, m, n, i and p pass the test for $(3, 3)$-diversity according to Definition 11, but leaf nodes d, e, g and o fail the test. Non-leaf nodes are not tested yet, but their node information have been computed. The total number of classes is $m = 5$. Consider node b, which has 5 children d, e, f, g and h. The node information of

b is computed as follows. According to Equation 2, the portion of tuples of each class is: $hit_{S_1} = (d.hit + h.hit) + \frac{1}{m-1} \times (e.miss + f.miss + g.miss) = (3+3) + \frac{1}{4} \times (4+9+8) = 11.25$, $hit_{S_2} = 11$, $hit_{S_3} = 9.75$, $hit_{S_4} = 11$, and $hit_{S_5} = 7$. Thus according to equation 3, $b.class = S_1$, $b.hit = \lceil 11.25 \rceil = 12$, and $b.miss = 50 - 12 = 38$.

4.2 Selecting Nodes to Merge

During the bottom-up level traversal, if some child of a non-leaf node fails the anonymity test, one of the failed children will be chosen to merge with some of its siblings. ATP uses heuristic rules to select siblings, so that their merge with the failed child will not only improve the privacy, but also preserve the classification accuracy.

Specifically, if one or more child fails the anonymity test, the one having the maximum $hit/(hit + miss)$ will be selected for merging. Such a node has the weakest protection of privacy and its pruning is likely to improve the privacy more quickly.

Siblings of the failed child is selected one at a time according to following rules.

- Rule 1: Select a sibling whose merging with the failed child will result in a new leaf node that satisfies the privacy requirement; else,
- Rule 2: Select a sibling which also fails the anonymity test; else
- Rule 3: Select a sibling which has the minimum $hit + miss$.

These rules are motivated by the desire of preserving the structure of the tree as much as possible. Intuitively, Rule 1 avoids merging too many siblings if the merging with one sibling can satisfy the privacy requirement; Rule 2 avoids merging too many siblings by handling two failed children with one merge; and Rule 3 reduces the impact of a merge on classification accuracy by selecting the sibling with the fewest tuples. If more than one node satisfies the same rules, ATP will choose a node according to a priority among the rules, as illustrated by the following example.

Example 16. Consider the decision tree in Figure 7. Recall that this tree contains node information calculated after the depth-first traversal and all leaf nodes have been tested against the $(3, 3)$-diversity anonymity requirement. There are total of 5 classes. The tree has 4 levels with the root being in level 1. The level-by-level traversal starts in level 3. The non-leaf nodes f and h only have *pass* children, so they are also marked *pass*. The non-leaf node j has one *fail* child, o, which is merged with its only sibling, p, which incidentally satisfies the heuristic Rule 1. The merge of nodes o and p result in a new node that *passes* the anonymity test and is also the only child of j. Thus, node j is replaced by the merged node.

In the next level, level 2, the non-leaf node b has 5 children among which nodes d, e, and g *fail* while nodes f and h pass the anonymity test. Since d has the highest $hit/(hit+miss) = 3/3 = 1$ among failed children, it is chosen to be merged. To find a sibling to merge with d, all remaining siblings are considered. Among these nodes, e, f and g satisfy Rule 1. In addition, e and g are also *fail* nodes. Finally, between e and f, node e has smaller $hit+miss$. Thus, e is chosen to merge with d, resulting in a new node represented by d, which satisfies $(3, 3)$-diversity. Now, the only failed child left is g. The remaining siblings are nodes f, h, and the new node d. All three of them satisfy Rule 1, but h has the smallest $hit + miss$. The merge of g and h gives the new node represented by g, which satisfies $(3, 3)$-diversity. At this time, all the leaf nodes in the decision tree *pass* the anonymity test. The final $(3, 3)$-diversified decision tree returned by the algorithm is shown in Figure 3, which is ready for pseudo-data generation.

5 Pseudo-data Generation From Decision Trees

In this Section, we focus on the problem of generating pseudo-data from a decision tree. We assume that (1) original-data tables of different data sources form a horizontal partition[5], that is, they have the same set of attributes; (2) an individual may have tuples at multiple data sources; (3) personal identifiers are not in published decision trees, thus also not in any pseudo-data table; and (4) both the local and the global knowledge models are decision trees.

5.1 Quality of Pseudo-Data

Since the pseudo-data is used for data mining, the quality of the learned knowledge models can indicate the quality of the pseudo-data. In our framework, there are two levels of pseudo-data, the global and the local levels. At the global level, the quality of global pseudo-data may be measured by the quality difference between the learned and the base global knowledge models. However, such a measure is not practical because it is not allowed to use the original-data to learn the base global knowledge model.

To solve this problem, we measure the quality of pseudo-data at the local level. Intuitively, a decision tree captures the most significant patterns (of joint distribution) of the training data and ignore minor patterns. We consider a pseudo-data table to be high-quality if it preserves the significant patterns that are in the original-data. Since in a multiple data source scenario, those significant patterns in the global original-data table are also likely to be significant in one or more local original-data tables, a high-quality global pseudo-data table is more likely to result from integrating high-quality local pseudo-data tables[6]. This intuition leads to the following decision tree pseudo-data generation problem.

> Given a decision tree h. Find a pseudo-data table D, so that if h' is a decision tree learned from D, the difference between classification accuracy of h and h', measured using a common testing set, is as small as possible.

Notice that there is no difficulty to obtain decision trees h and h': h is published by the data source and h' can be learned from a pseudo-data table. Since h is learned from a local original-data table, the similarity between h and h' can indicate the similarity between the local original-data and the local pseudo-data tables.

5.2 Generating Data From Paths

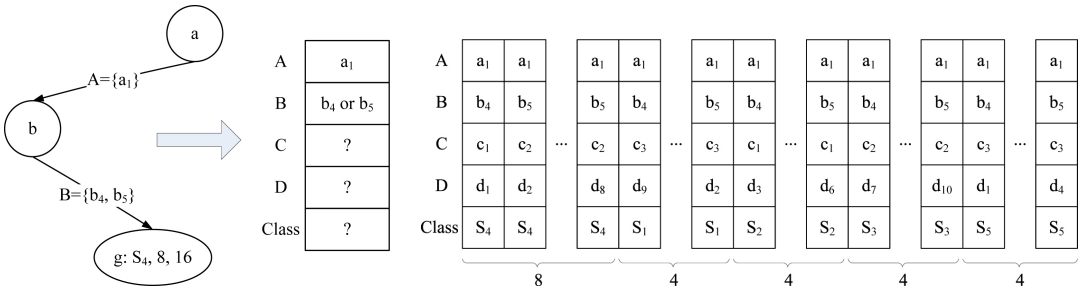
We start with an important property of an optimal pseudo-data table, where the optimum is due to the identical structure of the decision trees.

Definition 17. Two decision trees h and h' have identical structure if every path of h is a path of h' and vice verse.

Proposition 18. *Let h be a published decision tree, D be a pseudo-data table generated from h, and h' be a decision tree learned from D. If h and h' have identical structure, then*

[5] In a distributed environment, original-data tables of data sources can form a horizontal, vertical, or hybrid partition, where tables of a horizontal partition have a common set of attributes and tables of a vertical partition have different sets of attributes.

[6] The global pseudo-data table simply contains all the tuples in local pseudo-data tables, since they have the same set of attributes and no identifiers.



(a) Template (b) Pseudo tuples

Figure 8: Pseudo-data generation for one path

1. *every tuple in D satisfies exactly one path of h;*

2. *D contains exactly $p.hit + p.miss$ tuples that satisfy each path p in h.*

Proof. Since D is the training set of h', each tuple of D must satisfy exactly one path of h', therefore, it also satisfies the same path of h because h and h' have identical structure. Consequently, the hit and miss counts of each path in h' report exactly the number of tuples in D that satisfy the path in h', thus it is also the number of tuples in D which satisfy the same path in h. □

According to Proposition 18, paths of a published decision tree can be used as templates to generate pseudo tuples. Each pseudo tuple has the same attributes of the local original-data table, excluding the identifier. The values in a pseudo tuple can be determined as follows. If an attribute appears in the path, the value is determined according to the path label. If the attribute is absent from the path, the value can be determined according to some heuristics (see Section 5.3).

Example 19. Consider the path p labeled by $(A = \{a_1\} \wedge B = \{b_4, b_5\})$ in Figure 3. This path defines the template tuple in Figure 8(a). The values of attributes A and B are specified by the path label. Notice that the value for B is not determined yet, but it must be either b_4 or b_5. Since attributes C and D are absent from the path, their values are not determined (indicated by question marks). As for the class attribute, the majority class is S_4, and the non-majority classes can include all other values in the domain of the class attribute.

It is difficult to determine the optimal values of a pseudo tuple for attributes that are absent from the given path. Let us consider a simple case. Assume that a published decision tree contains s paths, where each path has r edges labeled with distinct attributes and is responsible of generating k pseudo tuples. Suppose each tuple has m attributes and each attribute has exactly K distinct values. Since a pseudo table may contain duplicate tuples, the k pseudo tuples of a path can be generated independently. Similarly, since paths of the decision tree partition tuples into disjoint groups, pseudo tuples of different paths can also be generated independently. Therefore, there are $N = ((K^{m-r})^k)^s$ possible pseudo datasets each satisfies Proposition 18. To find the optimal pseudo dataset by an exhaustive search can be prohibitively expensive. To the best of our knowledge, there is no efficient algorithm for finding an optimal pseudo table according to path-defined templates.

Algorithm Path-based Pseudo-data Generation (PGEN)
Input: A decision tree T and a set of attributes $\mathcal{A}$ of data
Output: A beg S of labeled tuples in $\mathcal{A}$
Method:
1. $S = \emptyset$;
2. for each path p of T do
3. for pseudo tuple t to be generated by p do
4. $t =$ a new empty tuple;
5. for each attribute $A \in \mathcal{A}$ do
6. if $A \in attr(p)$ then
7. $t[A] =$ a value selected from $value(A, p)$;
8. else $t[A] =$ a value selected from domain of A;
9. $t[class] =$ a class selected based on counts of p;
10. append t to S;
11. return S;

Figure 9: Path-based Pseudo-data Generation Algorithm

5.3 A Heuristic Algorithm

The generation of a pseudo-data table can be viewed as some reverse engineering of decision tree learning. In decision tree learning, at each tree node, the algorithm chooses the attribute which if used to split the training data (or tuples) in the node, will lead to the highest class purity as measured by information gain, Gini-index, or miss-classification rate. To reverse the process, the pseudo-data generation algorithm needs to guarantee that non-split attributes in the pseudo-data table have lower class purity than split attributes, so that if a decision tree learning algorithm is applied to the pseudo-data table, it will select the same split attributes as it does with the original-data. Thus, one heuristic is to assign random values to non-split attributes from a uniform distribution. Intuitively, although the distributions of data in split attributes of the original-data table are unknown, this heuristic can help minimize the class purity of non-split attributes in the pseudo-data table.

Based on this heuristic, the path-based pseudo-data generation (PGEN) algorithm (in Figure 9) uses each path of a decision tree to generate a bag of tuples (with possible duplicates). In line 2, each path is considered independently. The number of tuples to be generated by a path (line 3) can be determined in several ways. For example, it can be the sum of the hit and miss counts of the path, or a proportion of a user specified table size. In line 4, an empty template tuple is created with an unspecified value for each attribute. Subsequently, each attribute A that appears in the path p is assigned a value uniformly selected from $value(A, p)$ (line 7). Recall that $value(A, p)$ can contain multiple values for a categorical attribute and multiple intervals for a numeric attribute (see Section 3.2). Therefore, if A is categorical, one value is chosen randomly from these specified values. If A is numeric, we first randomly selected an interval and then select a value within the interval. This value can be a fixed default value, such as the middle point or an end point of the interval, or a random value drawn from the interval according to a given distribution (e.g., a uniform distribution). To each attribute that is absent from the path, we uniformly assign a random value from its domain (line 8). Finally, in line 9, a class label is assigned by a random draw according to the hit/miss ratio of the path. If the class should be a non-majority class, one of the non-majority classes will be randomly assigned to the tuple, according to a known

or a uniform distribution.

The time complexity of this algorithm is $O(|V| + |E| + N)$, where V and E are the sets of nodes and edges in the tree T, respectively, and N is the size of pseudo-data table.

Example 20. Figure 8(b) shows the pseudo tuples generated from the same path p in Figure 8(a). Assume there are 3, 10 and 5 distinct values in the domain of C, D and $Class$, respectively. According to node g, 24 tuples satisfy path p, therefore, 24 pseudo tuples need to be generated. In these tuples, attribute A always have a_1, attribute B is uniformly distributed between b_4 and b_5, and attributes C and D are assigned values uniformly randomly chosen from their respective domains. The classes of these tuples are assigned randomly according to the hit/miss ratio. Specifically, eight pseudo tuples are in class S_4, and the 16 remaining pseudo tuples are evenly distributed in the four non-majority classes.

6 Alternative Implementation Techniques

In previous sections, we presented techniques for implementing the knowledge model sharing framework (see Figure 1). In this section, we briefly discuss several existing techniques that may provide alternative implementation to several parts of the framework. The evaluation of these alternative implementation techniques is given later in Section 7.

6.1 Publishing Local Anonymized-Data

In our implementation of the framework, each data source publishes a local anonymized decision tree, from which a pseudo-data is generated. An alternative to having data miner to generate pseudo-data from published knowledge models is for each data source to publish an anonymized-data. Many data publishing methods have been reported in the literature. In this section, we briefly review three state-of-the-art data anonymization algorithms: the Incognito algorithm [11] with (c, ℓ)-diversity as the privacy measure, the one-dimensional algorithm [23] with simple ℓ-diversity as the privacy measure, and the K-Optimize algorithm [28] with ℓ-diversity as the privacy measure and the classification metric [29] as the utility measure.

6.1.1 Incognito Algorithm

The Incognito algorithm was originally proposed in [20] for implementing k-anonymity. It was adapted in [11] to implement (c, ℓ)-diversity.

The algorithm uses a graph to represent a set of generalization strategies (simply referred to as generalizations). Each generalization is a node in the graph and an edge between two nodes indicates that one node is a generalization of the other. Each generalization is encoded as a vector whose dimensions tell how to generalize some QI attributes. For example, vector $< A1, B2, C0 >$ represents a generalization strategy that for each tuple t, generalizes $t[A]$ by one level (i.e., replacing it with its parent), $t[B]$ by two levels (i.e., replacing it with its grandparent), and $t[C]$ by zero level (i.e., no generalization). Given a generalization, the data table can be partitioned and tested to determine if it satisfies the privacy requirement. In the graph, vectors have the same number of attributes. An edge (v, v') indicates that vector v and vector v' are identical except that v' is one level higher than v in exactly one attribute.

Given a data table and a set of taxonomies of attributes. The Incognito algorithm finds all generalizations that satisfy a given privacy requirement (either k-anonymity or (c, ℓ)-diversity) and does it efficiently by leveraging several properties of generalizations in ways similar to the Aproiri algorithm used in association rule mining [30, 31]. Starting from single-attribute generalizations that satisfy the privacy requirement, the algorithm iteratively constructs generalizations of more attributes. In iteration i, the algorithm performs a special breadth-first search the graph for i-dimensional generalizations that satisfies the given privacy requirement. These i-dimensional generalizations are then used to create a new graph of a set of candidate $(i+1)$-dimensional generalizations. This process continues until no new candidate generalization can be created. Among all generalizations found by the algorithm, the least general one will be used to anonymize the data table.

6.1.2 One-Dimensional Algorithm

Instead of focusing on generalization strategies, the 1-D algorithm for ℓ-diversity [23] works directly on data tuples. The main idea is to partition tuples into buckets of different SA values and to order them by one-dimensional values mapped from their multi-dimensional QI values.

The mapping of multi-dimensional QI values to one-dimensional numeric values is based on techniques, such as the Hilbert space-filling curve [32] or iDistance [33]. To facilitate the mapping, values of categorical attributes are first mapped to numeric values according to their positions in the in-order traversal of their taxonomy trees. After QI values are mapped, tuples are placed into buckets, so that, all tuples in one bucket has one SA value and are in an ascending order of their mapped QI values.

Next, the algorithm forms groups of ℓ tuples by repeatedly selecting tuples with smallest mapped QI values from the buckets. For each group, the algorithm chooses a new QI value that is minimally more general than those appear in the group and uses it to replace QI values of tuples of the group.

6.1.3 K-Optimize Algorithm

Unlike Incognito and One-Dimensional algorithms which both publish data for general purpose, the K-Optimize algorithm [28] publishes data for building predictive models, and as such, it optimizes the classification metric (CM) .

The algorithm assumes a total order over all values of all attributes and represents an anonymization succinctly as a sequence of intervals represented by their least values. For example, if the data has two attributes with 4 domain values and 6 domain values, respectively, then the anonymization <1, 4, 5, 7> is to partition the domain of the first attribute into {[1, 2, 3], [4]} and that of the second attribute into {[5, 6], [7, 8, 9, 10]}. With this formulation, the algorithm represents the entire space of all anonymizations as a set-enumeration tree. Each node in the tree has a head set representing the part of anonymization that has been determined and a tail set representing the part of anonymization yet to be determined. For example, a node {2}<3,4> can lead to children nodes {2, 3}<4> and {2,4}<>. If value 3 in the tail set can be pruned, this node will only lead to {2,4}<>.

The algorithm starts with a fully generalized dataset in which all tuples are identical and systematically specializes the dataset into one that is minimally k-anonymity (or ℓ-diversity, as implemented in our experiments). The algorithm does this in a depth-first traversal of the set-enumeration tree and performs cost-based pruning and dynamic tree rearrangement. At each node, it computes a CM cost using the head set and tries to prune

the tail set. A value in the tail set can be pruned if its inclusion into the head set will lead to violation of the privacy requirement or will not lead to any descendant with a lower CM cost. After pruning of the tail set, children of the node will be created and the process continues. Finally, the node with the least CM cost is chosen to create the anonymized-data.

6.2 Learning Anonymized Decision Trees

In our implementation of the framework, each data source uses the tree pruning technique in Section 4 to publish an anonymized decision tree. An alternative is to let each data source learn an anonymized decision tree directly from its original-data. The algorithm in [16] learns a k-anonymous decision tree by extending the well-known ID3 decision tree induction algorithm [34], so that for each candidate split point of a decision tree node, it not only computes the information gain of attributes, but also check if the resulting nodes satisfies k-anonymity. It selects the split point that has the highest information gain and satisfies the k-anonymity. This algorithm can be extended (as we did in our experiments) to learn an ℓ-diversified decision tree by enforcing ℓ-diversity instead of k-anonymity at each step of the learning process.

It is worth noting that the partition of data performed by this algorithm is essentially the same as those performed by a number of data anonymization algorithms, such as [22, 35, 36]. Although the latter produces anonymized-data, the patterns preserved by the these methods are no much different.

6.3 Ensemble Classifiers

In machine learning, an ensemble classifier aggregates a number of base classifiers to achieve a classification accuracy much better than can any single base classifier. In our PPDM framework, an ensemble classifier built from published local decision trees (as base classifiers) may be an alternative to learning global decision tree from pseudo-data.

The basic idea of the ensemble method is to use the set of base classifiers as a committee. Since the base classifiers complement each other, if a few committee members make incorrect classifications, the probability is high that the committee as a whole can still classify correctly. The two most popular ensemble methods are boosting [37] and bagging (or bootstrap aggregating) [17].

A basic boosting algorithm iteratively learns weak classifiers from weighted training data. Initially, all training tuples are equally weighted. After a weak model is learned from the data, it is used to adjust the weights of training tuples. Weights of tuples that are misclassified will be increased and others decreased. The data with adjusted weights is used to learn another weak classifier, and the weights of the training data are adjusted again based on how they are classified by the set of weak classifiers learned so far. The process ends when no new weak model can be learned. To classify an unseen tuple, the learned weak classifiers will each cast a vote and these votes are aggregated according a weighted formula that reflects the significance of member classifiers in producing accurate prediction.

A bagging algorithm assigns equal weights to votes of all member classifiers. Given a training set, a bagging method creates a set of new training sets by uniform sampling with replacement. Each new training set can be as large as the original training set and is used to learn a base classifier. Since the distribution of tuples in these training sets are usually different, the base classifiers learned from them are typically different. These base classifiers form a committee. The most popular prediction of base classifiers will be used as the prediction of the committee for unseen data.

In general, boosting has a better classification accuracy than bagging. However, in the multiple source environment we consider, data sources are autonomous and will not share data with each other. Thus, there is no global training set for a boosting method to learn weak models. This is why we do not consider boosting as an alternative method.

7 Empirical Study

In this section, we report results from an empirical study of our method of PPDM that uses the anonymity-guided decision tree pruning (ATP) and the path-based pseudo-data generation (PGEN).

7.1 The Setup of Experiments

In our experiments, we compared our ATP+PGEN method with a number of alternative methods that utilize techniques discussed in Section 6 to learn global knowledge models. We consider these methods because they all use ℓ-diversity including both the (c, ℓ)-diversity and simple ℓ-diversity, as the privacy measure. Although there exist many other privacy measures, due to the lack of a universal standard, methods using other privacy measures cannot be compared to our method nor to each other.

Our study is focused on two issues. One issue is the effectiveness of our pseudo-data generation algorithm in supporting the learning of the global knowledge model. We study this issue by comparing our PGEN algorithm with a naive pseudo-data generation algorithm and with the ensemble learning method. Another issue is the performance of different implementations of PPDM approaches. We studied this issue by comparing four implementations: ATP-PGEN, which uses our decision tree pruning algorithm ATP to publish local ℓ-diversity decision trees and uses our PGEN algorithm to generate pseudo-data for mining the global knowledge models; LDM-PGEN, which uses an anonymized decision tree learning algorithm LDM adapted from [16] to publish local ℓ-diversity decision trees and uses our PGEN method to generate pseudo-data for mining the global knowledge models; LDD, which uses data publishing methods [11, 23] to publish various types of ℓ-diversified data for mining the global knowledge models; and OPTI, which is adapted from [28] to publish ℓ-diversified data optimized for classification metric [29] for mining the global knowledge models. All four implementations are programmed in Java.

Datasets. We performed extensive experiments using five datasets (listed in Table 1) from the UCI Machine Learning Repository [38]. For the experiments, we removed missing values from the datasets. Although all five datasets were used in the study, for most of the experiments, we report results obtained from the Adult and the Nursery datasets. Results obtained from other datasets are similar.

datasets	#Instances	# Attributes		#Classes
		Nominal	Numeric	
Adult	45204	9	6	14
Car	1728	7	0	4
CMC	1473	8	2	3
Nursery	12960	9	0	5
Train	3000	9	1	2

Table 1: Experimental datasets

Data sources. To evaluate the impact of different data distributions, we considered a varying number of data sources ranging from one to ten. To account for the randomness in the data, we created data sources in a way similar to a ten-fold validation. Specifically, we divided a dataset randomly into ten subsets of equal sizes. We run each experiment ten times. In each run, we chose a different subset as the testing set for the learned global predictive model. The remaining nine subsets were combined together, which was then used to learn a base global predictive model, and subsequently distributed randomly and evenly to the given number of data sources as their local original-data. We then executed the run five times using these data sources. The results reported in this section is the average over all the passes (of all the runs).

Types of knowledge models. To evaluate PGEN and ATP, we compared the qualities of predictive models learned from both the original-data and the corresponding pseudo-data. We used decision tree (either pruned or un-pruned) as the local knowledge model for pseudo-data generation. For the (base or learned) global knowledge model, we used a variety of predictive models, including decision tree [34], Naive Bayes classifier [39, 40], and conjunctive rules [41]. These predicative models are chosen because they are widely used in practice and can be learned from our datasets. These predictive models were learned using public-domain Java implementations of their respective learning algorithms given in [34, 39, 40, 41].

Quality measures. We used several utility measures in our experiments. First, we used the *classification accuracy* and the *class match* to measure the quality of global predicative models. The classification accuracy is the fraction of testing tuples that are correctly classified and the class match is the fraction of testing tuples that are classified identically by the corresponding base and learned predictive models. Second, when decision trees were used as the global predicative model, we also measured *path match*, the fraction of paths shared by the base and the learned decision trees, as given by

$$ss(h, h') = \frac{|paths(h) \cap paths(h')|}{|paths(h) \cup paths(h')|}$$

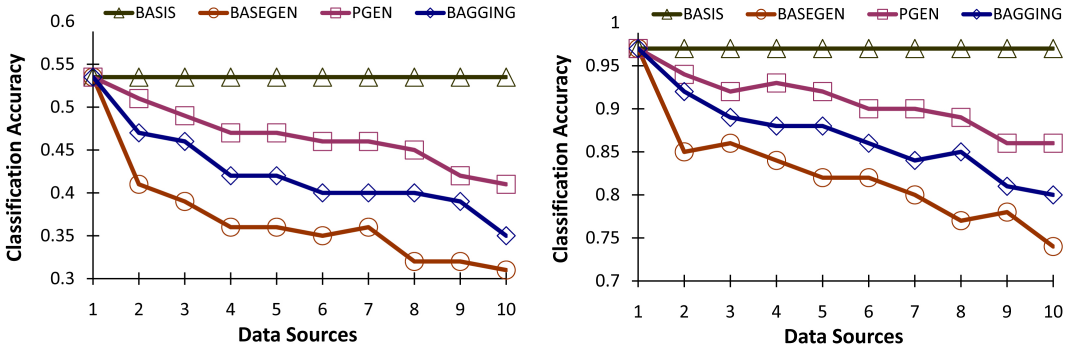
where $paths(h)$ is the set of paths in decision tree h. Third, we also used the classification metric (CM) [29, 42] to measure the quality of the anonymized-data obtained by data publishing methods and the pseudo-data generated from the published local knowledge models.

7.2 Quality of Pseudo-Data Generation

In this set of experiments, we compared three methods: PGEN, BASEGEN, and BAGGING, where PGEN is our path-based pseudo-data generation method (see Section 5.3); BASEGEN is a baseline pseudo-data generation method, which uses the published decision tree to assign class labels to a set of random tuples uniformly sampled from the data space; and BAGGING is an ensemble method [17] that uses published decision trees as base classifiers (see Section 6.3). To focus on pseudo-data generation, the published local decision trees were not pruned.

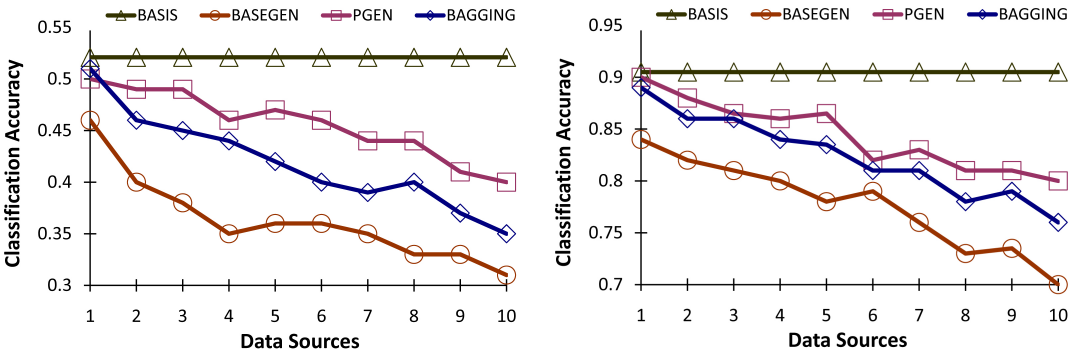
In the figures shown in this section, the x-axis is for the number of data sources, ranging from 1 to 10. The y-axis is for the measure of quality, which depending on the figure can be average classification accuracy, class match or path match. We also include the performance of the base global decision tree (referred to as the BASIS).

Figure 10a shows the classification accuracy of the methods with decision tree as the global predictive model. The results show that if there are two or more data sources, decision



(i) Adult Dataset (ii) Nursery Dataset

(a) Decision Trees



(i) Adult Dataset (ii) Nursery Dataset

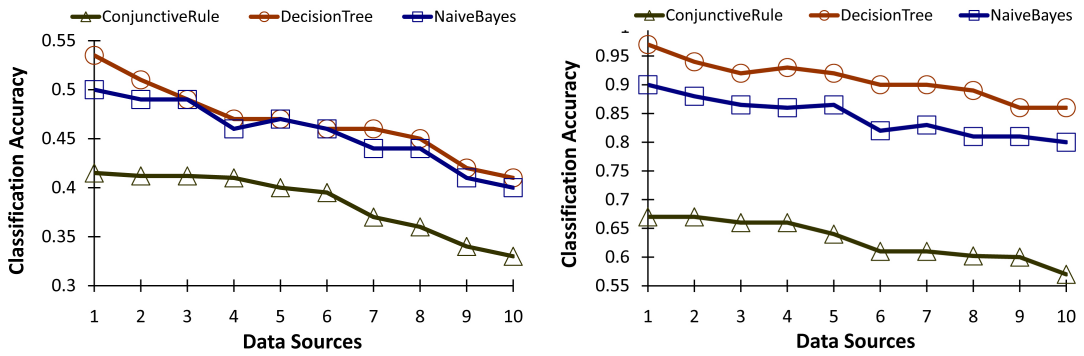
(b) Naive Bayes Classifiers

Figure 10: Classification Accuracy of One Global Knowledge Model

trees learned from pseudo-data generated by PGEN have higher classification accuracy than either decision trees learned from data generated by BASEGEN or ensemble classifiers built by BAGGING. If there is only one data source, no pseudo-data needs to be generated (because the published decision tree is already the desired output) and all three methods have the same classification accuracy. Notice that the classification accuracy of BASIS, the best that can be achieved by a global decision tree given the dataset, is not affected by the number of data sources. Figure 10a also shows that the classification accuracy of PGEN, BASEGEN and BAGGING decrease as the number of data sources increases. Similar trends can also be observed in Figure 10b, where the Naive Bayes Classifier is the global predictive model.

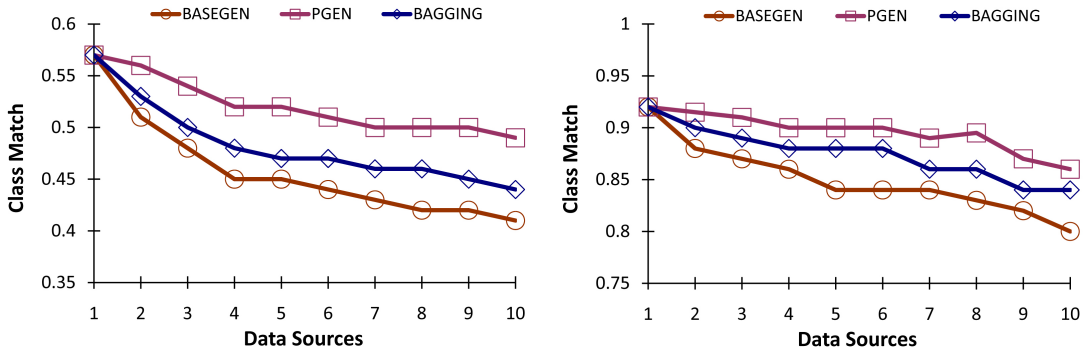
Although the classification accuracy of PGEN decreases, it still performs much better than BASEGEN. This is because when producing pseudo-data, BASEGEN creates tuples randomly and uses a given decision tree to assign class labels. Thus, the distributions of the pseudo-data and the original-data may be very different. As the number of data sources increases, this difference becomes increasingly larger. On the other hand, PGEN preserves the significant patterns in local original-data as indicated by the local decision trees.

PGEN also performs better than BAGGING. This is because as the number of sources increases, each local classifier is built from a smaller dataset and becomes more biased.



(i) Adult Dataset (ii) Nursery Dataset

Figure 11: Classification Accuracy of Multiple Global Knowledge Models



(i) Adult Dataset (ii) Nursery Dataset

Figure 12: Class Match of Global Decision Trees

Thus, it becomes more and more difficult for BAGGING to reach a consensus. However, PGEN can reduce the bias because the pseudo-data generated from the local decision trees contain more information, such as the non-majority class labels, and therefore, the quality of the global pseudo-data is still quite good.

Figure 11 shows the classification accuracy of several types of global predictive models learned from the pseudo-data generated by PGEN. Shown in the results, the average classification accuracy of global decision trees are better than that of global conjunctive rules and global Naive Bayes classifiers. This is expected since the published local predictive model is also decision tree. Because of this, we use only the decision tree as the base and the learned global knowledge models in the remaining of this subsection.

Figures 12 and 13 show results of using class match and path match, respectively, as the quality measure. In Figure 12, the trends of class match is very similar to that of classification accuracy. Again, PGEN outperforms both BASEGEN and BAGGING. BASIS is not shown here because its class match is always 1.

Figure 13 shows that decision trees learned from pseudo-data may have significantly different structures from those learned from local original-data. The figure shows only up to five data sources to prevent the chart from being clotted. In Figure 13, the two types of trees share less than 32% of paths and this ratio frequently drops blow 1%. Because the classi-

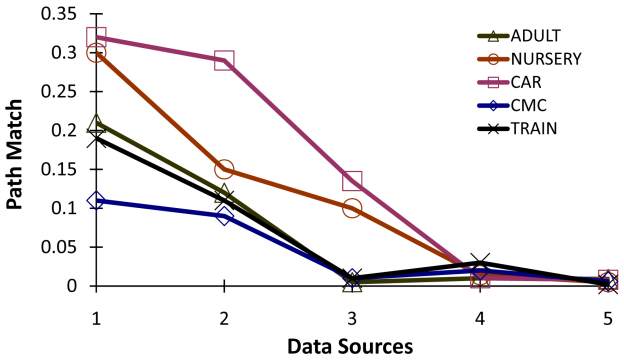


Figure 13: Path Match of Global Decision Trees on Multiple Datasets

fication accuracy is often the most important quality measure in practice for a predictive model (such as a decision tree), this experiment, performed only for PGEN, suggests that the quality of pseudo-data is not directly affected by the structural similarity of decision trees. This is another reason why we can prune decision trees for anonymization without losing too much classification accuracy.

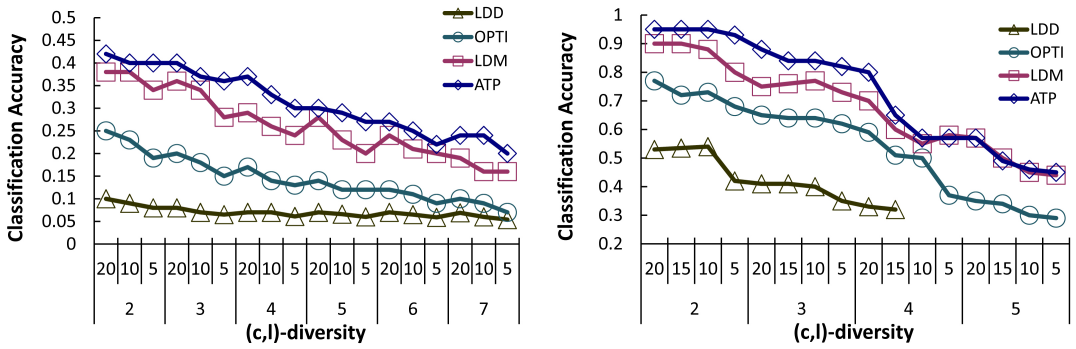
7.3 Effect of Anonymity-Guided Tree Pruning

In this set of experiments, we compared four methods: ATP, LDD, OPTI and LDM, where ATP is the decision tree pruning method (see Section 4), LDD is a data publishing method proposed in [11] for (c, ℓ)-diversity or in [23] for simple ℓ-diversity (see Section 6.1.1 and 6.1.2), OPTI is another data publishing method [28] for both types of ℓ-diversity (see Section 6.1.3), and LDM is the specialized decision tree learning algorithm [16] which learns an ℓ-diversity decision tree directly from the local original-data (see Section 6.2). In these experiments, decision tree and Naive Bayes Classifier were used as the global knowledge model and the classification accuracy and classification metric [29] were used as the utility measures.

We considered two scenarios: single data source and multiple data sources. In the single data source scenario, LDD and OPTI were used to generate anonymized-data and a traditional learning algorithm was used to learn decision tree from the anonymized-data. Both ATP and LDM were used to obtain anonymized decision trees. For the classification accuracy, we compared these decision trees. For the classification metric, we used PGEN to generate pseudo-data from anonymized decision trees, then compared the pseudo-data with the anonymized-data. In this case, we refer the two algorithms as ATP-PGEN and LDM-PGEN, respectively.

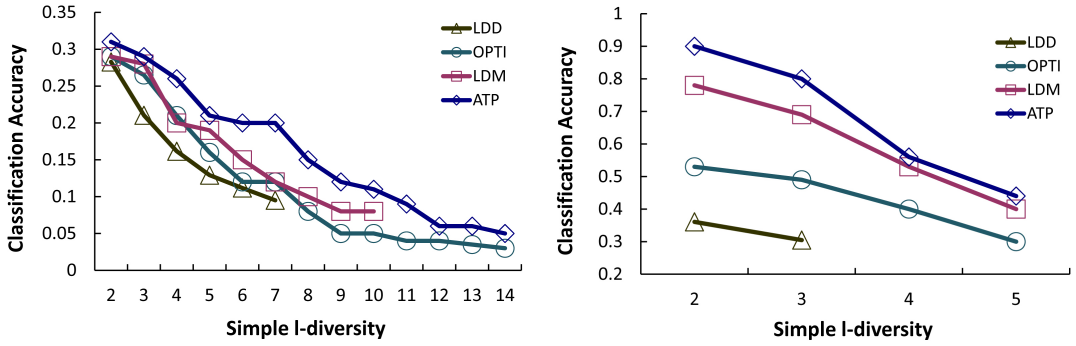
In the multiple data source scenario, LDD and OPTI were used to publish local anonymized-data by data sources and a traditional learning algorithm was used to learn the global knowledge model from the combined anonymized-data. ATP and LDM were used to publish local decision trees and PGEN was used to generate pseudo-data for learning global knowledge models (thus they are referred as ATP-PGEN and LDM-PGEN, respectively). Again, these global knowledge models were compared on their classification accuracy, and combined anonymized-data and global pseudo-data were compared on their classification metric.

We performed the experiments with a range of ℓ-diversity privacy requirements. For



(i) Adult Dataset (ii) Nursery Dataset

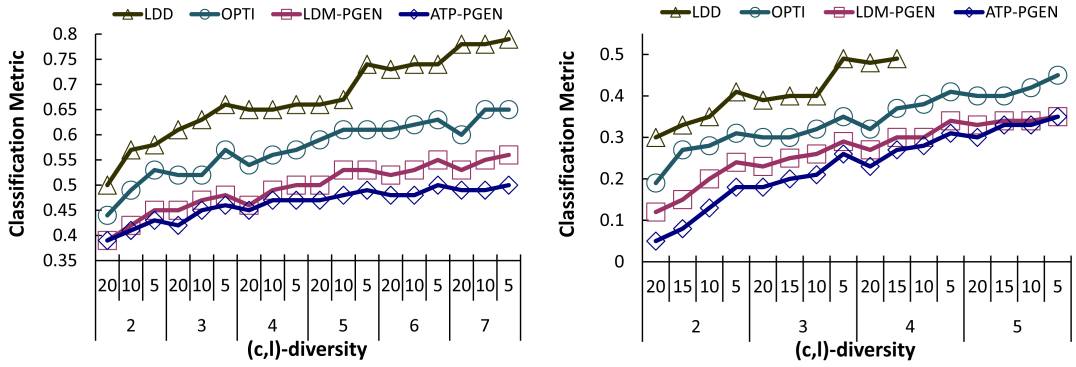
(a) (c, ℓ)-diversity



(i) Adult Dataset (ii) Nursery Dataset

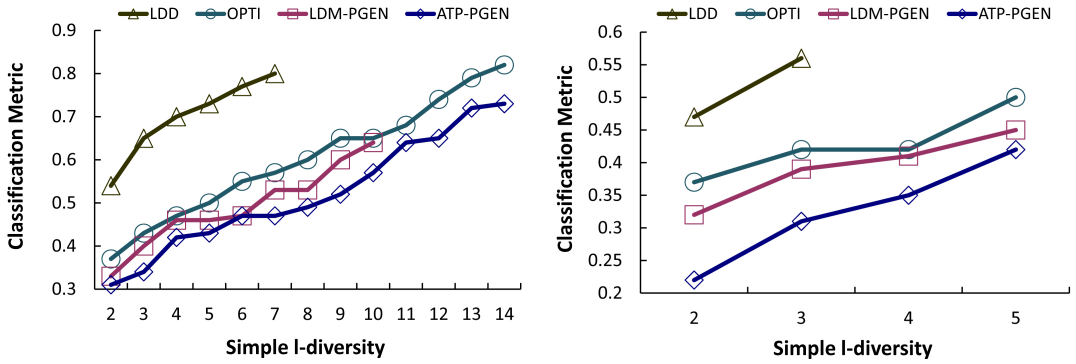
(b) Simple ℓ-diversity

Figure 14: Accuracy vs. ℓ-diversity of Decision Trees with Single Data Source



(i) Adult Dataset (ii) Nursery Dataset

(a) (c, ℓ)-diversity



(i) Adult Dataset (ii) Nursery Dataset

(b) Simple ℓ-diversity

Figure 15: Classification Metric vs. ℓ-diversity of Decision Trees with Single Data Source

Adult dataset, the range of simple ℓ-diversity was $2 \leq \ell \leq 14$, and the range of (c, ℓ)-diversity was $2 \leq \ell \leq 14$ and $c \in \{5, 10, 20\}$. For Nursery dataset, the range of simple ℓ-diversity was $2 \leq \ell \leq 5$ and the range of (c, ℓ)-diversity was $2 \leq \ell \leq 5$ and $c \in \{5, 10, 15, 20\}$.

In the figures presented in this section, the x-axis is privacy requirements ordered from left to right according roughly to the increasing strength of privacy requirements. The y-axis is the average classification accuracy of global decision trees, or the classification metric of global pseudo-data and combined anonymized-data. For readability, results on Adult dataset are plotted for $2 \leq \ell \leq 7$.

Figure 14a shows classification accuracy of ATP, LDM, OPTI and LDD in the single data source scenario with (c, ℓ)-diversity as privacy measure and decision tree as the global knowledge model. As shown in the results, ATP performs slightly better than LDM and much better than OPTI and LDD. A similar trend is also shown in Figure 14b, where the simple ℓ-diversity is the privacy measure. Since pseudo-data generation was not involved in these experiments, the results clearly indicate that anonymity-guided tree pruning effectively balances privacy and utility.

The results also show that LDD and LDM could not satisfy some privacy requirements

that can be satisfied by ATP and OPTI. For example, as shown in Figure 14b(i), LDD cannot satisfy privacy requirements when $\ell > 7$ and LDM fails at $\ell = 10$. LDD fails because the original-data does not satisfy the eligibility conditions [18], and LDM fails because it could not find a decision tree in which all the children of the root satisfy the privacy requirement. On the other hand, both OPTI and APT can satisfy all the privacy requirements we tested because they make more effort and finer adjustment. For example, OPTI is able to explore the entire space of anonymizations with its depth-first traversal and can suppress tuples. Similarly, APT can prune branches little by little, by merging a bad node with a near-by sibling.

Figure 15a shows classification metric of ATP-PGEN, LDM-PGEN, OPTI and LDD in the single data source scenario with (c, ℓ)-diversity as privacy measure[7]. Since PGEN randomly generates values for attributes that do not appear in the paths of anonymized decision trees, we do not consider such attributes in the pseudo-data for a fair comparison. Similarly, the attributes which are neither QI nor SA are also not considered in the anonymized-data. As shown in the results, ATP-PGEN has less penalties than LDM-PGEN, OPTI and LDD. A similar trend is also shown in Figure 15b, where the simple ℓ-diversity is the privacy measure. These results indicate that the pseudo-data generated from our anonymized decision trees captures the significant patterns of the original-data better than other methods.

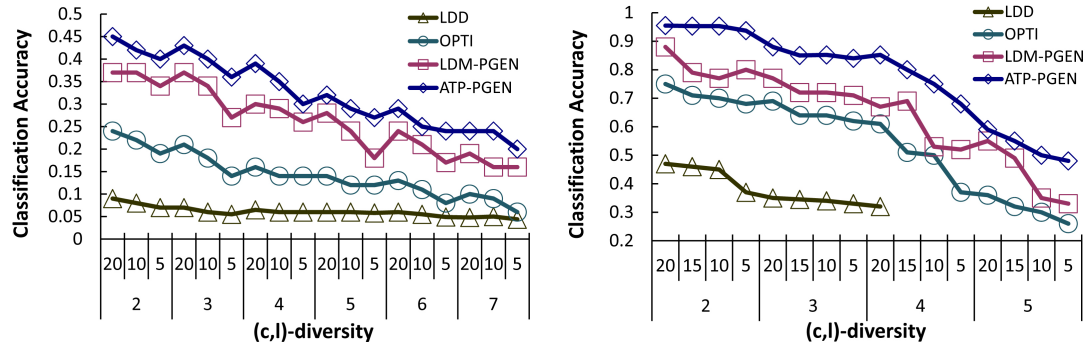
Figure 16a shows classification accuracy of ATP-PGEN, LDM-PGEN, OPTI and LDD in the multiple data source scenario (with 10 data sources) using the (c, ℓ)-diversity as the privacy measure and decision tree as the global knowledge model. As shown in the results, ATP-PGEN clearly outperforms both LDM-PGEN and LDD over all privacy requirements we tested. In addition, the performance differences in Figure 16a are larger than in Figure 14a. For example, in Figure 14a(ii), ATP is about 10% better than LDM, 20% better than OPTI, and 40% better than LDD, but in Figure 16a(ii), ATP-PGEN is about 15% better than LDM-PGEN, 25% better than OPTI, and 50% better than LDD. A similar trend is also found in Figure 16b, which uses Naive Bayes as the global knowledge model. Figure 16c also shows that ATP-PGEN is better than other methods measured by the classification metric.

Relative to ATP (resp. ATP-PGEN), the poorer performance of LDM (resp. LDM-PGEN) may be because that during the learning of the tree, LDM may abandon some crucial paths. For example, if LDM were used to learn the decision tree in Figure 5, a split on attribute B at node b would not be valid because it would result in children nodes d, e, and g not satisfying the given $(3, 3)$-diversity requirement. If at node b, there is no valid split on attributes C and D either, there will be no path going out b. Using ATP, however, it is possible to combine nodes d and e into a new node and nodes g and h into another new node, so that, the split on attribute B at node b is still valid.

One reason for LDM-PGEN to perform worse than LDM is that the anonymized decision trees used by PGEN to generate pseudo-data typically have lower classification accuracy. Another reason is that the increasing number of data sources can add more noises into the global pseudo-data.

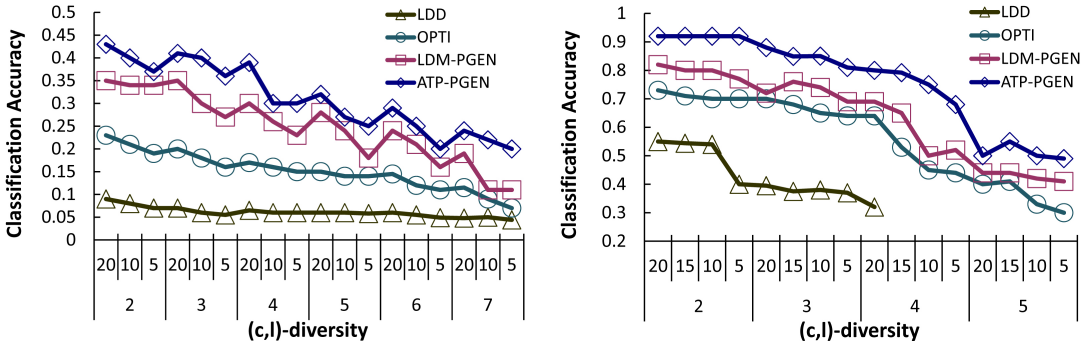
A reason for ATP-PGEN to perform much better than LDD may be that PGEN pays special attention to preserve statistical features important to decision tree learning, but LDD does not. On the other hand, OPTI performs better than LDD perhaps because it optimizes the classification metric. However, optimizing the classification metric somewhat contradicts with ℓ-diversity because it tries to increase the frequency of the majority class in equiva-

[7] We did not use the general information loss metric (LM) or discernibility metric (DM) [42] as utility measures to compare these methods. This is because PGEN always creates the most specific values (pseudo or real), thus has no general information loss. Using LM will be unfair to LDD and OPTI. On the other hand, DM is only suitable for k-anonymity, not for ℓ-diversity.



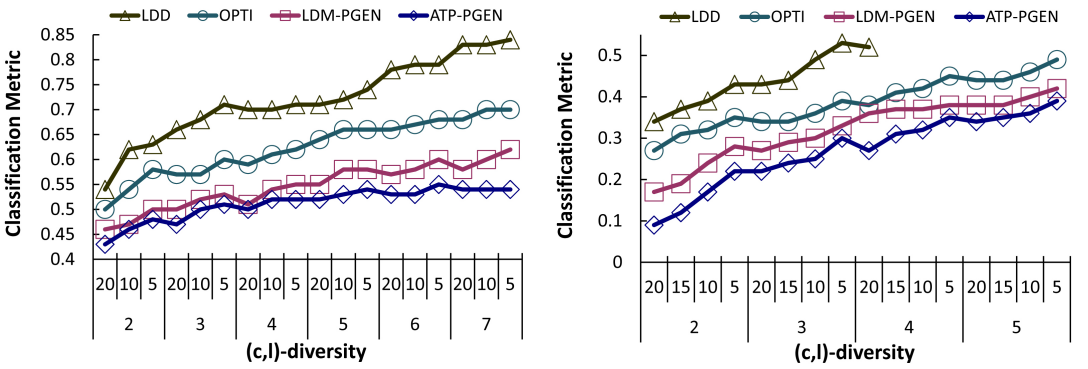
(i) Adult Dataset (ii) Nursery Dataset

(a) Decision Trees



(i) Adult Dataset (ii) Nursery Dataset

(b) Naive Bayes Classifiers



(i) Adult Dataset (ii) Nursery Dataset

(c) Classification Metric

Figure 16: Quality vs. (c, ℓ)-diversity of Knowledge Models with 10 Data Sources

lence classes, but the ℓ-diversity tries to prevent a single class from being dominant in an equivalence class. Thus at certain point (e.g., ℓ is high), both the diversity of classes in equivalence classes and the suppression of tuples (to satisfy the privacy requirement) can lead to very high penalties. This may explain why OPTI performs worse than ATP-PGEN.

It is worth noting that the pseudo-data generated by ATP-PGEN from a decision tree that satisfies an ℓ-diversity requirement may not itself satisfy the same ℓ-diversity requirement. This is because that paths in the decision tree may not have all QI attributes. Therefore pseudo tuples generated from the same path may belong to different equivalence classes, causing some equivalence class to fail the ℓ-diversity. However, there is no privacy concern because the pseudo-data does not describe real persons and the source of its generation, the anonymized decision tree, satisfies the privacy requirement.

8 Related Work

Cryptographic PPDM methods have been proposed to learn various types of (global) knowledge models, including decision trees [2, 3], naive Bayes classifiers [43], association rules from horizontally [44] and vertically [45] partitioned data, clusters from horizontally [46, 47] and vertically [48] partitioned data, and collaborative filtering based prediction [49]. All these methods may produce knowledge models that could disclose private information [1].

In the pioneer work in [5], data is perturbed by adding random noises and decision trees are learned by estimating the original-data distribution. This method was later found to have a serious flaw [50]. In addition to decision trees, perturbation have also been used to produce data for learning clusters [51] and association rules [6, 8, 52, 53]. Other perturbation methods include matrix perturbation [7, 8], ρ_1-to-ρ_2 privacy breaching [6], minimality attack [26], and differential privacy [15]. The utility of data produced by these perturbation methods is usually low because instances in the published data will no longer represent the original information.

Using generalization to protect privacy was first studied in [10], which proposed k-anonymity as a privacy measure. Efficient implementations of k-anonymity include global recoding methods, such as single-dimension [20], multiple-dimension [22], bottom-up [54], top-down [28, 35] and genetic methods [29], and local recoding methods, such as [24, 55]. The ℓ-diversity was proposed in [11, 18, 23] to prevent homogeneous and background attacks. There are several variants of ℓ-diversity including (α, k)-anonymity [56] and (ϵ, m)-anonymity [57]. In addition, a number of generalization methods used other privacy measures, including t-closeness [21], m-invariance [58], BCF-anonymity [59], personalized privacy [19], (c, k)-safety [60] and privacy skyline [25]. To improve the utility, it was proposed in [61] to publish both the anonymized and the marginal tables. Our work is within the framework of generalization, but differs from existing generalization methods in two aspects. First, we consider an environment of multiple data sources rather than single data source; second, we measure privacy of knowledge models rather than that of datasets.

The publishing of a set of clusters satisfying k-anonymity and ℓ-diversity was proposed in [13, 62]. The published model includes the QI value of the cluster center, the size of the cluster, and the sensitive values in a cluster. Publishing a set of contingency tables satisfying some requirements of differential privacy was proposed in [14, 63]. These contingency tables are marginals containing sanitized incomplete joint distribution of the original-data. These studies gave theoretic treatment of the problems, but did not provide efficient algorithms for practical use. On the other hand, [64] proposed a method to publish a safe view that can be used to easily obtain a naive Bayes classifier satisfying ρ_1-to-ρ_2 privacy measure,

and [16] presented a decision tree learning algorithm that incorporates ℓ-diversity to learn an anonymized decision tree. These methods can only be applied to single data source.

Various pseudo-data generation methods were proposed before. A method was proposed in [65] to generate synthetic data that statistically mimics the original-data for a specific application while providing privacy guarantees. The method in [66] generates pseudo-data from statistics of condensed groups, where each group contains a fixed number of data records closest to a randomly selected private record in a multidimensional space. The method in [13] generates pseudo-data according to information of a given number of clusters learned from the original-data. Unlike these methods, our method generates pseudo-data according to path information contained in decision trees.

Ensemble methods, such as boosting [37] and bagging [17], were extensively studied in data mining to learn global knowledge models. These methods need a testing set to resolve conflicts among base classifiers, which is impractical in a multiple source environment when privacy is a concern. Our method is to learn global knowledge models from pseudo-data rather than to use published local models as base classifiers. This not only helps to protect privacy, but also has flexibility to learn other types of knowledge models rather than classifiers. A recent work [67] has the similar idea to merge local patterns into a global pattern.

There were several studies on secure sharing of association rules and frequent patterns [68, 69], which focus on hiding sensitive association rules and blocking inference channels. Unlike these studies, we protect sensitive information, such as disease or total assets of individuals, in data. Besides the data having the explicit sensitive attributes, there are also many studies on the privacy in other types of data, such as transactions [70] and social network graphs [71].

9 Conclusions and Future Work

In this paper, we propose a knowledge model sharing based approach for privacy-preserving data mining. We present a framework in which each data source publishes an anonymized knowledge model learned from its own data, and a global knowledge model is learned from the pseudo-data generated from the published local knowledge models. For publishing anonymized knowledge models, we adapt the k-anonymity and ℓ-diversity as the privacy measure for decision trees, and present a heuristic algorithm that prunes a decision tree to satisfy a given anonymity requirement. For pseudo-data generation, we present a heuristic algorithm that generates pseudo-data according to paths of a decision tree. We empirically compare our method with several PPDM methods that utilize existing techniques, including three methods that publish anonymized-data, one method that learns anonymized decision trees directly from the original-data, and one method that uses ensemble classification. Our results show that in both single data source and multiple data source environments and for several different datasets, predictive models, and utility measures, our method can obtain significantly better predictive models (especially decision trees) than the other methods.

There are several interesting directions of future study. For example, the disclosure of privacy by knowledge models is still an open problem. Privacy measures for knowledge models other than decision trees do not yet exist and it is not clear whether and how other existing privacy measures for data, such as differential privacy [15], can be adapted for knowledge models. Also, although pseudo-data may provide flexibility to learn global knowledge models that are different from published local models, it is yet unclear for given

types of local and global knowledge models that are not decision trees, what pseudo-data generation method should be used to preserve data characteristics that critically affect the quality of the global knowledge model.

References

[1] Patrick Sharkey, Hongwei Tian, Weining Zhang, and Shouhuai Xu. Privacy-preserving data mining through knowledge model sharing. In *ACM SIGKDD International Workshop on Privacy, Security, and Trust in KDD*, pages 97–115, 2007.

[2] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology - Crypto 2000*, pages 36–54, 2000.

[3] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations*, 4(2):12–19, 2002.

[4] Murat Kantarcioglu, Jiashun Jin, and Chris Clifton. When do data mining results violate privacy? In *International Conference on Knowledge Discovery and Data Mining*, pages 599–604, 2004.

[5] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *ACM SIGMOD International Conference on Management of Data*, pages 439–450, 2000.

[6] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaching in privacy preserving data mining. In *ACM Symposium on Principles of Database Systems*, pages 211–222, 2003.

[7] Jim Dowd, Shouhuai Xu, and Weining Zhang. Privacy-preserving decision tree mining based on random substitutions. In *International Conference on Emerging Trends in Information and Communication Security*, pages 145–159, 2006.

[8] Shipra Agrawal and Jayant R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *IEEE International Conference on Data Engineering*, pages 193–204, 2005.

[9] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *International Conference on Very Large Data Bases*, pages 901–909, 2005.

[10] P. Samarati and L. Sweeney. Protecting privacy when disclosing information:k-anonymity and its enforcement through generalization and suppression. In *IEEE Symposium on Research in Security and Privacy*, pages 188–206, 1998.

[11] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. In *IEEE International Conference on Data Engineering*, page 24, 2006.

[12] Lijie Zhang and Weining Zhang. Generalization-based privacy-preserving data collection. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 115–124, 2008.

[13] Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. In *ACM Symposium on Principles of Database Systems*, pages 153–162, 2006.

[14] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems*, pages 273–282, 2007.

[15] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages and Programming*, pages 1–12, 2006.

[16] Arik Friedman, Ran Wolff, and Assaf Schuster. Providing k-anonymity in data mining. *The International Journal on Very Large Data Bases*, 17(4):789–804, 2008.

[17] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[18] Xiaokui Xiao and Yufei Tao. Anatomy: Simple and effective privacy preservation. In *International Conference on Very Large Data Bases*, pages 139–150, 2006.

[19] Xiaokui Xiao and Yufei Tao. Personalized privacy preservation. In *ACM SIGMOD International Conference on Management of Data*, pages 229–240, 2006.

[20] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: Efficient fulldomain k-anonymity. In *ACM SIGMOD International Conference on Management of Data*, pages 49–60, 2005.

[21] Ninghui Li and Tiancheng Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *IEEE International Conference on Data Engineering*, pages 106–115, 2007.

[22] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *IEEE International Conference on Data Engineering*, page 25, 2006.

[23] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. Fast data anonymization with low information loss. In *International Conference on Very Large Data Bases*, pages 758–769, 2007.

[24] Yang Du, Tian Xia, Yufei Tao, Donghui Zhang, and Feng Zhu. On multidimensional k-anonymity with local recoding generalization. In *IEEE International Conference on Data Engineering*, pages 1422–1424, 2007.

[25] Bee-Chung Chen, Kristen LeFevre, and Raghu Ramakrishnan. Privacy skyline: Privacy with multidimentional adversarial knowledge. In *International Conference on Very Large Data Bases*, pages 770–781, 2007.

[26] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *International Conference on Very Large Data Bases*, pages 543–554, 2007.

[27] Hongwei Tian and Weining Zhang. Extending l-diversity for better data anonymization. In *International Conference on Information Technology: New Generations*, pages 461–466, 2009.

[28] Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *IEEE International Conference on Data Engineering*, pages 217–228, 2005.

[29] V. Iyengar. Transforming data to satisfy privacy constraints. In *International Conference on Knowledge Discovery and Data Mining*, pages 279–288, 2002.

[30] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *International Conference on Very Large Data Bases*, pages 487–499, 1994.

[31] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *International Conference on Very Large Data Bases*, pages 161–180, 1995.

[32] Bongki Moon, H.V. Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the clustering properties of the hilbert space-filling curve. In *IEEE Transactions on Knowledge and Data Engineering*, pages 124–141, 2001.

[33] Rui Zhang, Panos Kalnis, Beng Chin Ooi, and Kian-Lee Tan. Generalized multidimensional data mapping and query processing. *ACM Transactions on Database Systems*, 30(3):661–697, 2005.

[34] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[35] Benjamin C. M. Fung, Ke Wang, and Philip S. Yu. Top-down specification for information and privacy preservation. In *IEEE International Conference on Data Engineering*, pages 205–216, 2005.

[36] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Workload-aware anonymization. In *International Conference on Knowledge Discovery and Data Mining*, pages 277–286, 2006.

[37] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[38] A. Asuncion and D.J. Newman. The uci machine learning repository. http://mlearn.ics.uci.edu/MLRepository.html, 2007.

[39] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:2–3, 1997.

[40] David J. Hand and Keming Yu. Idiot's bayes-not so stupid after all? *International Statistical Review*, 69:385–398, 2001.

[41] van Zyl Jacobus and Cloete Ian. Heuristic functions for learning fuzzy conjunctive rules. In *2004 IEEE international conference on systems, man and cybernetics*, pages 2332–2337, 2004.

[42] M. Ercan Nergiz and Chris Clifton. Thoughts on k-anonymization. *Data and Knowledge Engineering*, 63:622–645, 2007.

[43] Murat Kantarcioglu and Chris Clifton. Assuring privacy when big brother is watching. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 88–93, 2003.

[44] Murat Kantarcioglu and Chris Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

[45] Jaideep Vaidya and Chris Clifton. Privacy preserving association rule mining in vertically partitioned data. In *International Conference on Knowledge Discovery and Data Mining*, pages 639–644, 2002.

[46] S. Jha, L. Kruger, and P. McDaniel. Privacy preserving clustering. In *European Symposium On Research In Computer Security*, pages 397–417, 2005.

[47] Ali Inan, Selim V. Kaya, Yucel Saygin, Erkay Savas, Ayca A. Hintoglu, and Albert Levi. Privacy preserving clustering on horizontally partitioned data. *Data and Knowledge Engineering*, 63(3):646–666, 2007.

[48] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *International Conference on Knowledge Discovery and Data Mining*, pages 206–215, 2003.

[49] Ibrahim Yakut and Huseyin Polat. Privacy-preserving hybrid collaborative filtering on cross distributed data. *Knowledge and Information Systems*, 30:405–433, 2012. 10.1007/s10115-011-0395-3.

[50] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *IEEE International Conference on Data Mining*, pages 99–106, 2003.

[51] Srujana Merugu and Joydeep Ghosh. Privacy-preserving distributed clustering using generative models. In *IEEE International Conference on Data Mining*, pages 211–218, 2003.

[52] A. Evmievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *International Conference on Knowledge Discovery and Data Mining*, pages 343–364, 2002.

[53] Shariq J. Rizvi and Jayant R. Haritsa. Maintaining data privacy in association rule mining. In *International Conference on Very Large Data Bases*, pages 682–693, 2002.

[54] Ke Wang, Philip S. Yu, and Sourav Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *IEEE International Conference on Data Mining*, pages 249–256, 2004.

[55] Jian Xu, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi, and Ada Wai-Chee Fu. Utility-based anonymization using local recoding. In *International Conference on Knowledge Discovery and Data Mining*, pages 785–790, 2006.

[56] Raymond Chi-Wing Wong, Jiuyong Li, Ada Wai-Chee Fu, and Ke Wang. (α, k)-anonymity: An enhanced k-anonymity model for privacy-preserving data publishing. In *International Conference on Knowledge Discovery and Data Mining*, pages 754–759, 2006.

[57] Jiexing Li, Yufei Tao, and Xiaokui Xiao. Preservation of proximity privacy in publishing numeric sensitive data. In *ACM SIGMOD International Conference on Management of Data*, pages 473–486, 2008.

[58] Xiaokui Xiao and Yufei Tao. m-invariance: Towards privacy preserving re-publication of dynamic datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 689–700, 2007.

[59] Benjamin C. M. Fung, Ke Wang, Ada Wai-Chee Fu, and Jian Pei. Anonymity for continuous data publishing. In *International Conference on Extending database technology: Advances in database technology*, pages 264–275, 2008.

[60] David J. Martin, Daniel Kifer, Ashwin Machanavajjhala, Johannes Gehrke, and Joseph Y. Halpern. Worst-case background knowledge for privacy-preserving data publishing. In *IEEE International Conference on Data Engineering*, pages 126–135, 2007.

[61] Daniel Kifer and J. E. Gehrke. Injecting utility into anonymized datasets. In *ACM SIGMOD International Conference on Management of Data*, pages 217–228, 2006.

[62] Jian Li, Ke Yi, and Qin Zhang. Clustering with diversity. In *International Colloquium on Automata, Languages and Programming*, pages 188–200, 2010.

[63] Shiva Prasad Kasiviswanathan, Mark Rudelson, Adam Smith, and Jonathan Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *ACM Symposium on Theory of Computing*, pages 775–784, 2010.

[64] Barzan Mozafari and Carlo Zaniolo. Publishing naive bayesian classifiers: privacy without accuracy loss. In *Proceedings of the VLDB Endowment*, volume 2, pages 1174–1185, 2009.

[65] Ashwin Machanavajjhala, Daniel Kifer, John Abowd, Johannes Gehrke, and Lars Vilhuber. Privacy: Theory meets practice on the map. In *IEEE International Conference on Data Engineering*, pages 277–286, 2008.

[66] C. Aggarwal and P. Yu. A condensation approach to privacy preserving data mining. In *International Conference on Extending Database Technology*, pages 183–199, 2004.

[67] Maria C. Gaya and J. Ignacio Giraldez. Merging local patterns using an evolutionary approach. *Knowledge and Information Systems*, 29:1–24, 2010. 10.1007/s10115-010-0332-x.

[68] S. Oliveira, O. Zaane, and Y. Saygin. Secure association rule sharing. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 74–85, 2004.

[69] Zhihui Wang, Wei Wang, and Baile Shi. Blocking inference channels in frequent pattern sharing. In *IEEE International Conference on Data Engineering*, pages 1425–1429, 2007.

[70] Grigorios Loukides, Aris Gkoulalas-Divanis, and Bradley Malin. Coat: Constraint-based anonymization of transactions. *Knowledge and Information Systems*, 28:1–32, 2010. 10.1007/s10115-010-0354-4.

[71] Xiaowei Ying and Xintao Wu. On link privacy in randomizing social networks. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 28–39, 2009.