kACTUS 2: Privacy Preserving in Classification Tasks Using k-Anonymity

Slava Kisilevich, Yuval Elovici, Bracha Shapira, and Lior Rokach

Department of Computer and Information Science
Konstanz University
Universitaets Strasse 10
Box 78, 78457 Konstanz, Germany
slaks@dbvis.inf.uni-konstanz.de[‡]
Department of Information System Engineering and
Deutsche Telekom Laboratories at Ben-Gurion University
Ben Gurion University, Be'er Sheva, 84105, Israel
{elovici,bshapira,liorrk}@bgu.ac.il[§ ¶ ‖]

Abstract. k-anonymity is the method used for masking sensitive data which successfully solves the problem of re-linking of data with an external source and makes it difficult to re-identify the individual. Thus k-anonymity works on a set of quasi-identifiers (public sensitive attributes), whose possible availability and linking is anticipated from external dataset, and demands that the released dataset will contain at least k records for every possible quasi-identifier value. Another aspect of k is its capability of maintaining the truthfulness of the released data (unlike other existing methods). This is achieved by *generalization*, a primary technique in k-anonymity. Generalization consists of generalizing attribute values and substituting them with semantically consistent but less precise values. When the substituted value doesn't preserve semantic validity the technique is called *suppression* which is a private case of *generalization*. We present a hybrid approach called *compensation* which is based on *suppression* and *swapping* for achieving privacy. Since *swapping* decreases the truthfulness of attribute values there is a tradeoff between level of swapping (information truthfulness) and suppression (information loss) incorporated in our algorithm.

We use k-anonymity to explore the issue of anonymity preservation. Since we do not use generalization, we do not need a priori knowledge of attribute semantics. We investigate data anonymization in the context of classification and use tree properties to satisfy k-anonymization. Our work improves previous approaches by increasing classification accuracy.

Keywords: anonymity, privacy preserving, generalization, suppression, data mining.

[‡] http://www.informatik.uni-konstanz.de/arbeitsgruppen/infovis/mitglieder/slava-kisilevich
[§] http://www.ise.bgu.ac.il/faculty/elovici
[¶] http://www.ise.bgu.ac.il/faculty/bracha-s
[‖] http://www.ise.bgu.ac.il/faculty/liorr

C.S. Gal, P.B. Kantor, and M.E. Lesk (Eds.): ISIPS 2008, LNCS 5661, pp. 63–81, 2009.

1 Introduction and Motivation

In today's computerized world, when storage device costs are low, data about individuals are extensively available and easily collected: Internet surfers leave all sorts of tracks at every visited site (from the computer's IP address to submitted forms with private data); email providers scan email traffic; medical patient records are stored in databases; governments maintain records about every citizen; private companies and organizations (such as travel agencies, flight and insurance companies, etc.) collect data about people from a variety of sources. What would happen if some of this information, perhaps financial or health-related, became publicly available? It could very possibly lead to community ostracism or even dismissal from work. Clearly, there is a growing concern among individuals that the data they share about themselves, either voluntarily or due to various regulations, should be protected.

Technological advances are increasing the demand for data and there is a growing interest for private companies and universities in using these data for research, to investigate patterns of behaviour and to draw conclusions. In the face of this continually expanding interest in exploiting available datasets, we have seen over the past 10 years increased attention in privacy preserving data mining, a field which tries to provide a tradeoff between maintaining privacy and using the private data.

As an example of this thin line between preserving privacy and using private data, consider a typical hospital's cardiology center where the database undoubtedly includes patient records comprising personal information (such as name, social security number); demographic data (race and age); and medical information (symptoms, diagnosis and medications). An external researcher is interested in using this database in order to generate a complications model for these patients with the aim of introducing a new treatment protocol. A patient record should, of course, be stripped of ifentifiable information before being shared with any person other than the patient's primary physician. Obviously unique identity fields such as name and social security number should be completely removed from the dataset. However, other fields can be used to disclose the real identity of the patient indirectly, especially if part of the data can be linked to other data sources that may include the patients' identities. For example, if there is only a single patient in the database who is 90 years old, then providing the age datum may reveal the identity of this patient.

A real-life example was demonstrated by Sweeney [1]. She purchased a copy of the Cambridge Massachusetts voter registration list and also obtained a copy of medical records of Massachusetts state emploees. Linking together the data from the two datasets, she managed to find medical information about the governor of Massachusetts. Only three attributes were needed: *ZIP, Birth date* and *Sex*. She demonstrated that even though sensitive data such as social security numbers were not present in any of the two datasets and examining only one of two datasets alone couldn't provide an inference about a particular individual, nevertheless, cross-linking several datasets provided the governor's personal medical records. In an effort to enhance data privacy, Sweeney proposed a new

method, k-anonymity which guarantees privacy protection via linking and allows the safe release of information without invalidating the truthfulness of an individual record.

In this paper we discuss privacy preservation and data mining problems in terms of classification. We propose an algorithm for privacy preserving data mining that performs dataset anonymization using the k-anonymity model while taking into account its effect on classification results. We extend the k-anonymity model by providing new definitions and use several anonymization techniques together in order to get better results in terms of accuracy than reported in the literature.

In Section 2 we discuss related work while Section 3 provides some definitions used in the article and in formulating the problem. In Section 4, we explain how our proposed algorithm works and Section 5 provides experimental results. Section 6 concludes the paper.

2 Related Work

k-anonymity is a popular approach to masking data and avoiding re-identification of sensitive data through common attributes [2]. A dataset complies with k-anonymity protection if individual data stored in the released dataset cannot be distinguished from at least $k-1$ individuals whose data also appears in the dataset. This protection guarantees that the probability of identifying an individual based on the released data in the dataset does not exceed $1/k$. Generalization and suppression are the most common methods used for de-identification of the data in k-anonymity based algorithms [3],[4],[5],[6]. Generalization is a process of substiting an attribute value with a more general value but which is semantically consistent with the previous one using so-called *attribute hierarchies*. One of the advantages of generalization over other disclosure limitation techniques is that it preserves the truthfulness of the information. However, a major drawback of existing generalization techniques is that domain hierarchy trees are required for every quasi-identifier attribute on which k-anonymity is to be applied [4],[5],[6],[7],[8],[9],[10]. These domain hierarchy trees are generated manually by the user before applying the generalization process. In order to overcome the major drawback of generalization we effectively use suppression in our algorithm.

While many other approaches for masking private data exist, we would like to mention *swapping* since it is used in our algorithm. Proposed in 1978 by Dalenius and Reiss [11], data swapping selectively modifies a fraction of the records in the database by exchanging a subset of attributes between selected pairs of records. The advantage of swapping lies in the fact that it introduces uncertainty for potential attackers about the true value of a sensitive attribute. Since 1978 many swapping techniques have been proposed [12],[13],[14]. A common characteristic to all of the techniques is the attempt to preserve the statistical characteristics of the original dataset (univariate distribution of the swapped attribute, natural dependency between fields, means and variances) rather than high classification results. In contrast to the original definition and purpose of

swapping, the technique that we propose here employs swapping as a means for achieving k-anonymity and as a retaining factor for the information loss incurred by suppression. To the best of our knowledge no research has been performed to date where several privacy-preserving techniques are mixed together.

In our previous research [15], we presented a k-anonymity classification tree-based suppression (kACTUS). kACTUS wraps a decision tree inducer which is used to induce a classification tree from the original dataset. kACTUS then uses this tree to apply k-anonymity to the dataset while minimizing the tradeoff between k-anonymity constraints and classification quality. The resultant anonymous dataset can then be sent to an external user that may utilize any induction algorithm for training a classifier over the anonymous dataset. kACTUS uses the suppression approach alone by performing a random selection of instances, which is known to reduce the quality of the data when inefficiently applied. Moreover this randomness injection may cause the performance of the algorithm to be unstable. Results might be improved if a greedy selection rule is used.

In this paper we present a variation of kACTUS that we refer to as kACTUS-2. Our new revised algorithm does not randomly select records, instead it uses an information gain criterion for selecting the best instances to suppress. Moreover, kACTUS-2 efficiently uses multi-dimensional suppression and swapping so that the performance of the new method outperforms previous algorithm and other generalization methods which require manually defined generalization taxonomies.

3 Problem Formulation

In this section several definitions which will be used later in the article are introduced and the problem formulation is presented (Definition 9).
The following two definitions are adopted from [3].

Definition 1 (Quasi-identifier). *Given a population U, a person-specific table $PT(A_1, \ldots, A_n), f_c : U \rightarrow PT$ and $f_g : PT \rightarrow U^{'}$, where $U \subseteq U^{'}$. A quasi-identifier of PT, written Q_T is a set of attributes $A_i, \ldots, A_j \subseteq A_1, \ldots, An$ where $\exists p_i \in U$ such that $f_g(f_c(p_i)[Q_{PT}]) = p_i$.*

Definition 2 (k-anonymity). *Let $RT(A_1, \ldots, A_n)$ be a table and Q_{RT} be the quasi-identifier associated with it. RT is said to satisfy k-anonymity if and only if each sequence of values in $RT[Q_{RT}]$ appears at least k times in $RT[Q_{RT}]$.*

Definition 3 (Decision tree). *Decision tree is a classifier consisting of decision and leaf nodes. A decision node specifies a test over one of the attributes. The best attribute is selected using some of the accepted metrics. A leaf node specifies a value of the target attribute. Decision nodes which are closer to the root were selected first thus they can be regarded as more important in terms of classification than nodes closer to leaves.*

kACTUS-2 receives a decision tree as input and works with its decision nodes. The algorithm does not use information about the value of the target node

directly from the decision tree, thus when a term *leaf node* will be further encountered it will denote only that the node is a decision node which have zero child nodes.

Definition 4 (Complying Node and Non-complying Node). *Let us suppose that we follow a tree branch from the root node a_1 to a leaf node a_n. We construct an attribute-value set going down the branch up to the leaf node a_n. If the number of tuples in a dataset, consisting of the attribute-value set is greater or equal to the k-anonymity threshold than we call a leaf a_n a complying node, otherwise we call it a non-complying node. From the properties of decision trees we can state that there are only complying and non-complying nodes in the tree.*

Definition 5 (Suppression). *As defined by Sweeney in [3], suppression means that a specific value from the original dataset will be substituted by a meaningless one in the anonymized dataset. In this paper, we assume that suppression is used on the non-complying node only where we prune the leaf node of the non-complying node, thus suppressing all the attribute values which are associated with that node.*

We used the question mark "?" as a symbol for the suppressed value which is interpreted by the evaluated inducers as a missing value.

Definition 6 (Compensation). *Using Definition 4 about complying and non-complying nodes, we propose a general approach for achieving k-anonymity using decision trees called compensation. A complying node can manipulate its associated records such that it can compensate part of its records in favor of a non-complying node in order to turn a non-complying node into a complying one. kACTUS-2 performs compensation by suppression and compensation by swapping. In addition compensation works only between sibling leaf nodes which have a common parent.*

Definition 7 (Compensation by Suppression). *We now extend the definition of suppression given in Definition 5. Compensation by suppression is always accompanied by suppression of non-complying nodes. First, we check how many records are required for the non-complying node to be compliant. Let us assume that the number of required records is $k - m$ where m is the number of records associated with the non-complying node. Suppose that there is a compliant node which can compensate $k - m$ of its records. We select $k - m$ records from a complying node (selection criteria are described in Section 4) and suppress their attribute values associated with the complying leaf node. The suppression of the non-complying node and compensation by suppression guarantee that a non-complying node will be compliant.*

Definition 8 (Compensation by Swapping). *First, we check how many records are required for the non-complying node to be compliant. Let us assume that the number of required records is $k - m$ where m is the number of records associated with the non-complying node. Suppose that there is a compliant node*

which can compensate $k-m$ of its records. We perform $k-m$ iterations such that on every iteration we check (using criteria explained in Section 4 what is the best class value that is required for the non-complying node. Then, we search for a record with the specified class value which is associated with the complying node. If such a record is found we swap its leaf node value with the leaf node of the non-compying node. If such a record is not found we select another record (using specific criteria explained in Section 4) and swap its leaf node value. It is clear that after $k-m$ iterations there will be k records associated with a non-complying node.

Definition 9 (Optimal k-anonymity Transformation for a Classification Problem). *Given k-anonymity threshold $\in [1, m]$, an inducer I, a dataset DS with input attribute set $A = a_1, \ldots, a_n$ and target class y from a distribution D over the labeled instance space, the goal is to find an optimal transformation such that S' satisfies k-anonymity. Optimality is defined in terms of minimizing the deterioration in the generalized accuracy over the distribution D as a result of replacing classifier I(S) with classifier I(S').*

4 Method

Recall from the problem definition 9 that our objective is to get an anonymous dataset where predictive performance of a classifier trained on the anonymous dataset will be comparable to the performance of the classifier trained on the original dataset. In our approach we wrap an existing classification tree induction algorithm (such as C4.5) trained on the quasi-identifiers of the original dataset. Having the classification model of the original dataset, we can perform k-anonymization by manipulating leaves of the decision tree. If all the leaves in the tree are complying nodes then we can say that the dataset complies with k-anonymity. For leaves that do not comply with k-anonymity we perform one of two operations: *Suppression and Compensation by Suppression* (Def. 5 and 7) or *Compensation by Swapping* (Def. 8).

4.1 How a Complying Node Is Selected for Compensation by Swapping

Every complying node is checked against a non-complying node. First, we perform a virtual *compensation by swapping* on a complying node and check what is the final entropy [16] of its remaining records. Shannon entropy of a random variable X having the probability distribution $p = (p_1, \ldots, p_n)$ is given by:

$$H(p_1, \ldots, p_n) = \sum_{i=1}^{n} -p_i log_2 p_i \tag{1}$$

Lastly, the complying node minimal final entropy is selected for compensation.

4.2 How Compensation by Swapping is Performed

Once a complying node is selected for compensation, we perform an actual swapping in the following way. We check which class value has less influence in terms of entropy on the records associated with a non-complying node. Given a class value we search in the original dataset for a record associated with the complying node having that class value. Provided such a record is found, we select it and swap its attribute value associated with a leaf node by the attribute value of the non-complying leaf node. If such a record was not found, we check which class value will have less influence on the final entropy after we remove such a record from the set of records associated with a complying node. The same swapping process as described above is applied on the selected record. Given initial number $m < k$ of records associated with the non-complying node, where k is the anonymity threshold, we perform $k - m$ iterations and in every iteration we perform swapping as described above.

4.3 The kACTUS-2 Algorithm

Algorithms 1-4 describes our new method for k-anonymity. Since the algorithm uses several helper functions which are quite straightforward, we don't provide their pseudo-code. However the general descriptions and explanations are presented below.

root - returns the root node of CT
height - returns the height (length of the longest path from the node to the leaf)
parent - returns parent of the node
count-instances - counts instances in the dataset associated with a particular node
move-complying-node - moves instances associated with the complying node to the anonymized dataset with non-quasi-identifiers of the original instance if any.
remove-leaf-nodes - remove leaf nodes of a node
get-total-instance-count - counts the total number of instances associated with all the nodes in the set
move-root-non-complying-nodes - moves the instances associated with root nodes by first suppressing all quasi-identifiers (the root node contains only one quasi-identifier) and keeps only the target class value of the original instance along with non-quasi-identifiers of the instance, if any.
get-non-complying-leafs - given a node, returns all leafs which don't comply with k-anonymity.
get-complying-leafs - given a node, returns all leafs which comply with k-anonymity.
move-instances - like *move-complying-node* but takes set of nodes as an input.
calculate-final-compliant-entropy - The explanation is given further in this section.

swap-from-complying - performs swapping of the attribute value of the complying leaf node to the attribute value of the non-complying leaf node and swapping of the class value required by the non-complying node to keep its entropy at a low level. See Definition 8 and Subsection 4.2 for detailed explanations.

move-non-complying-instances - like *move-instances* but before moving such instances, the attribute of the leaf node is removed from all the instances (suppressed).

compensate-from-complying - See Definition 7 for a detailed explanation.

The algorithm requires the following input parameters:

1. k-anonymity threshold KT
2. swapping threshold ST
3. original dataset OD
4. classification tree CT
5. set of non-quasi-identifiers

The output of the algorithm is the anonymized dataset AD.

The process is as follows: we iterate over the classification tree while it has at least one root node. Having selected a *root-node* we find the longest path from it to the leaf node. If the longest path is of a height greater than 1 it means that the *root-node* has children and thus we can call the main function called *PerformAnonymization* (line 1.15), otherwise we need to check how many instances are associated with the particular *root-node*. If the number of instances is greater or equal to the k-anonymity threshold then we move the instances from the original dataset (CS) to the anonymized dataset (line 1.18). After that we remove the *root-node* from the classification tree (line 1.19). If the number of instances is less than the k-anonymity threshold we add the node to a set of non-complying nodes (line 1.21). When all root nodes are pruned from the classification tree we need to check how many instances in total are associated with **all** the non-complying nodes stored in the non-complying set (line 1.25). If the total number of instances is greater or equal to the k-anonymity threshold we call the *move-root-non-complying-nodes* function (line 1.25) which moves these instances to the anonymized dataset. Only the target attribute and non-quasi-identifiers of the original instance are retained. On the other hand, if the total number of instances are less than the k-anonymity threshold we do not copy these instances to the anonymized dataset. In such cases the anonymized dataset can contain less instances than the original one, however data loss is bounded by the k-threshold, so in a worst case scenario the maximum number of records which can be lost is $k - 1$.

Algorithm 2 describes the *PerformAnonymization* procedure. This procedure gets the leaf node with the longest path to the root found in Algorithm 1 as a parameter. Using the *parent* function we acquire the parent of this node (line 2.2). Then, we check how many instances are associated with the parent. This means that we need to count all the instances of its child nodes. If the total number of instances is less than the k-threshold then we just prune the parent

node by removing all its children (line 2.4). In lines 2.7 and 2.8 we get two sets which contain complying and non-complying leaf nodes (children of the parent node). An additional check is performed on line 2.9 such that if the set which holds non-complying leafs is empty, there is no need to continue further. We just move all instances associated with the complying leaf node (line 2.10) and prune all the children (line 2.11). We call two functions *PerformSwapping* and *PerformSuppression* in succession on lines 2.14 and 2.15. Finally we prune all the children nodes (line 2.16).

Algorithm 3 describes the *PerformSwapping* procedure. It starts by iterating over non-complying nodes (lines 3.2-3.22). First, we calculate how many instances are required in order to make the non-complying nodes compliant (lines 3.3). Then, the *required-ratio* is calculated and compared to the swapping threshold, ST(line 3.4). We perform swapping only if the ratio of required instances is less than the predefined ST. For every non-complying node, the inner loop is started over complying nodes (lines 3.9-3.16) in order to find the best complying node to perform swapping (lines 3.11-3.15) Finally, when the *best-complying-node* is found, we move the instances associated with the non-complying node to the anonymized dataset using the *move-instances* function (line 3.20) and perform swapping using the *swap-from-complying* function (line 3.21).

Algorithm 4 describes the *PerformSuppression* procedure. In lines 4.2-4.13 we iterate over complying nodes. We need to check whether the complying node is capable of compensating some of its instances to non-complying nodes. In order to implement this action, we start the inner loop over non-complying nodes (lines 4.3-4.11). We obtain the number of instances which can be given by the complying node (line 4.4 and the number of instances required by the non-complying node (line 4.5). If the number of instances which the complying node can compensate is greater or equal to the number of required instances then compensation is possible, otherwise compensation is not used. Finally, we move instances associated with the non-complying node to the anonymized dataset (line 4.9) but with the leaf attribute value suppressed. After this step is carried out, the *compensate-from-complying* function is called to compensate the required number of instances of the complying node (line 4.10). The remaining instances of the complying node will be moved to the anonymized dataset (line 4.12).

5 Experimental Evaluation

This section presents the results of various issues that were examined in regard to the proposed algorithm: (1) verification of the proposed hybrid approach for achieving k-anonymity without reasonable loss of classification accuracy; (2) comparison of kACTUS-2 with TDS, TDR and kADET algorithms presented in [6],[9] and [10] in terms of classification accuracy, information loss and statistical significance; (3) comparison of kACTUS-2 with former algorithm kACTUS in terms of classification accuracy, information loss, statistical significance and data loss.

Algorithm 1. kACTUS-2 (Part 1)

```
   Input:
 1:  KT - k-anonymity threshold
 2:  ST - swapping threshold
 3:  OD - original dataset
 4:  CT - classification tree
 5:  NQI - non-quasi-identifier set

   Output:
 6:  AD - anonymous dataset

 7: procedure MAIN(KT, ST, OD, AD, CT, NQI)
 8:    CS ← OD                          ▷ work on copy of the original data set
 9:    AD ← ∅
10:    non-complying-node-set ← ∅
11:    for all root-node in root(CT) do
12:       while height(root − node) > 0 do
13:          longest-node ← get-longest-node(root-node)
14:          if height(longest-node) > 1 then
15:             PerformAnonymization(longest-node, KT, ST, CS, AD)
16:          else                          ▷ the longest node is the root node
17:             if count-instances(longest-node, CD) ≥ KT then
18:                move-complying-node(longest-node, CS, AD)
19:                remove-leaf-nodes(longest-node)
20:             end if
21:             non-complying-node-set ← non-complying-node-set ⋃ longest-node
22:          end if
23:       end while
24:    end for
25:    if get-total-instance-count(non-complying-node-set) ≥ KT then move-root-
   non-complying-nodes(non-complying-node-set,CS, AD, NQI)
26:    end if
27: end procedure
```

5.1 Datasets

A comparative experimental-study was carried out using five datasets from the UCI Repository [17]: Adult, German Credit, TTT, Glass Identification and Waveform. Researchers into k-anonymity mostly work with Adult and German Credit datasets as they are the most suitable, publicly available benchmarks. We use three additional publicly available datasets (TTT, Glass Identification and Waveform) to evaluate the performance of the above mentioned algorithms.

The Adult dataset has six continuous and eight categorical attributes and a binary class attribute which represents income levels (less or more than 50K). The numerical data was discretized by a supervised discretization filter using a Weka datamining framework [18] as a preprocessing step. The dataset has 45222 records. A domain generalization hierarchy was built using all 14 attributes and

Algorithm 2. kACTUS-2 (Part 2)

```
1: procedure PerformAnonymization(longest-node, KT, ST, CS, AD, NQI)
2:     parent-node ← parent(longest-node)
3:     if count-instances(parent-node,CD) < KT then
4:         remove-leaf-nodes(parent-node)
5:         return
6:     end if
7:     non-complying-leafs ← get-non-complying-leafs(parent-node)
8:     complying-leafs ← get-complying-leafs(parent-node)
9:     if non-complying-leafs = ⊘ then
10:        move-instances(complying-leafs, CS, AD, NQI)
11:        remove-leaf-nodes(parent-node)
12:        return
13:    end if
14:    PerformSwapping(complying-leafs,non-complying-leafs, KT, ST, CS, AD, NQI)
15:    PerformSuppression(complying-leafs, non-complying-leafs, KT, ST, CS, AD,
       NQI)
16:    remove-leaf-nodes(parent-node)
17: end procedure
```

was adopted from [6]. The domain generalization hierarchy was used by TDS, TDR and kADET algorithms.

The German Credit dataset contains observations on 30 variables for 1000 past applicants for credit. Each applicant was rated as a good credit (700 cases) or bad (300 cases). We used nine quasi-identifiers as was proposed by [9]: *credit history, savings account, duration in month, status of existing checking account, credit amount, purpose, property, installment rate in percentage of disposal income, other debtors.*

The TTT dataset encodes the complete set of possible board configurations at the end of tic-tac-toe games. It contains 958 instances and nine categorical attributes and *positive* and *negative* binary class values. All attributes were used as quasi-identifiers.

The Glass Identification database contains 214 instances and nine continuous attributes. We performed supervised discretization of continuous attributes. The class label represents seven types of glass: *building_windows_float_processed, building_windows_non_float_processed, vehicle_windows_float_processed*, *vehicle_windows_non_float_processed, containers, tableware, headlamps.* All attributes were used as quasi-identifiers.

The Waveform Database Generator contains 5000 instances and 40 continuous attributes which were discretized by Weka. Class label represents three classes of waves. All attributes were used as quasi-identifiers.

5.2 Comparing to Other *k*-Anonymity Algorithms

In order to evaluate classification accuracy of the proposed method and compare it with other algorithms, we divided each dataset into a so-called 5x2 *cross*

Algorithm 3. kACTUS-2 (Part 3)

```
1: procedure PerformSwapping(complying-nodes,non-complying-nodes, KT, ST, CS, AD, NQI)
2:     for all nc in non-complying-leafs do
3:         required ← KT - count-instances(nc)
4:         required-ratio ← required / KT
5:         if ST < required-ratio then
6:             continue                                  ▷ swapping is not possible
7:         end if
8:         best-complying-node ← ∅
9:         for all c in complying-leafs do
10:            if count-instances(c) - KT > required then
11:                entropy ← calculate-final-compliant-entropy(c,nc,KT)
12:                if entropy is minimal then
13:                    best-complying-node ← entropy
14:                end if
15:            end if
16:        end for
17:        if best-complying-node = ∅ then
18:            continue          ▷ no best complying node was found for swapping
19:        end if
20:        move-instances(nc, CS, AD, NQI)
21:        swap-from-complying(best-complying-node, nc, required, NQI)
22:    end for
23: end procedure
```

validation proposed by [19]. The method is based on five iterations of a two-fold cross validation. All five algorithms were evaluated on training sets with different k-anonymity thresholds. We evaluated four classification algorithms in the induction phase in order to examine various induction biases: C4.5 [20], PART [18], Naïve Bayes and Logistic [21]. All experiments were performed using Weka. The performance graphs we generated represent the classification accuracy as a function of k.

The evaluation results are very encouraging. Figures 1-5 present performance graphs using a C4.5 classifier. Figure 1 displays a classification accuracy graph for the Adult dataset with the anonymity threshold of $5 \leq k \leq 1000$. The classification accuracy of the original model is 85.96. The k values we used are: $5, 20, 50, 100, 500, 1000$. We can see that the classification accuracy of kACTUS and kACTUS-2 with $k = 5$ is considerably higher than that of TDS, TDR and kADET. But the accuracy of kACTUS-2 converges to the accuracy of TDS for $k = 1000$, while the classification accuracy of kACTUS becomes worse than kADET starting from about $k = 700$ and starting from $k = 900$ with TDS.

Figure 2 displays the classification accuracy of the German dataset. The k values we used are: $5, 10, 15, 20, 30, 40, 50, 80, 100$. The original classification accuracy is 71.67. Again, with a small value of $k = 5$, kACTUS-2 performs better than all other algorithms. With higher values of k, the performance of kACTUS-2

Algorithm 4. kACTUS-2 (Part 4)

```
1: procedure PerformSuppression(complying-nodes,non-complying-nodes, KT,
   ST, CS, AD, NQI)
2:     for all c in complying-leafs do
3:         for all nc in non-complying-leafs do
4:             can-compensate ← count-instances(c) - KT
5:             required ← KT - count-instances(nc)
6:             if can-compensate < required then
7:                 continue        ▷ complying node has less instances than required for
   compensation
8:             end if
9:             move-non-complying-instances(nc, CS, AD, NQI)
10:            compensate-from-complying(c,required)
11:        end for
12:        move-instances(c, CS, AD, NQI)
13:    end for
14: end procedure
```

gets worse and in the range $45 \leq k \leq 78$ the classification accuracy of kACTUS is higher. Classification accuracy of kACTUS-2 converges to the accuracy of kADET for $k = 100$ and both algorithms provide best results at $k = 100$.

Figure 3 shows the classification accuracy of the TTT dataset using the C4.5 classifier. The original classification accuracy is 81.20. We used the following k-anonymity thresholds: $5, 10, 15, 20, 30$. The performance of kACTUS-2 is better than all the other algorithms. Only when starting from $k = 25$, its performance decreases below the classification accuracy of kACTUS. However it

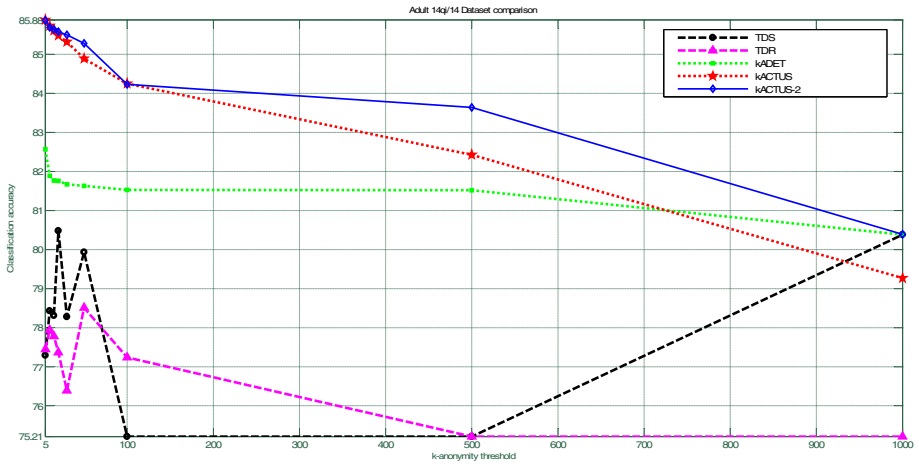

Fig. 1. Adult 14qi/14. Classification accuracy vs. k-threshold. C4.5 inducer.

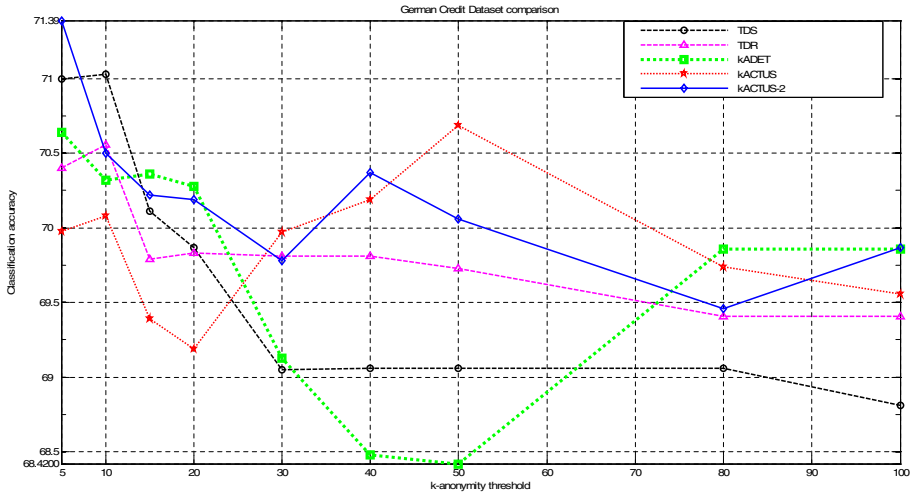

Fig. 2. German Credit. Classification accuracy vs. k-threshold. C4.5 inducer.

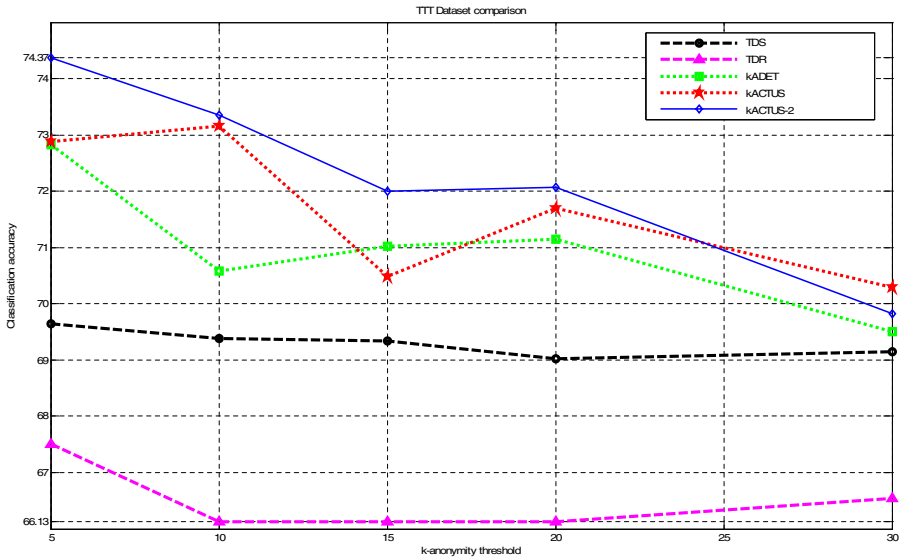

Fig. 3. TTT. Classification accuracy vs. k-threshold. C4.5 inducer.

remains higher than kADET, TDS and TDR. The worst performance is that of TDR.

From Figure 4, we can see that on small values of k, all algorithms except TDR behave almost the same with minor variations. kADET seems to perform better up to $k = 15$. On higher values of k kACTUS outperforms all the other

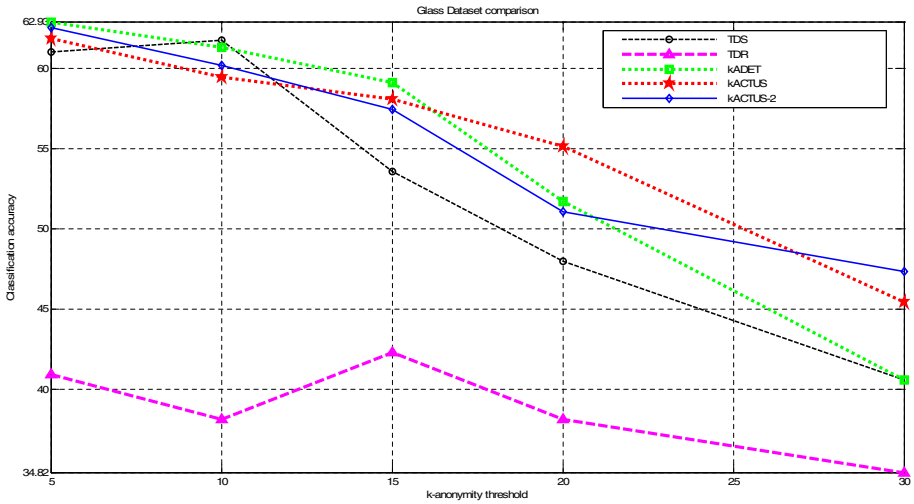

Fig. 4. Glass. Classification accuracy vs. k-threshold. C4.5 inducer.

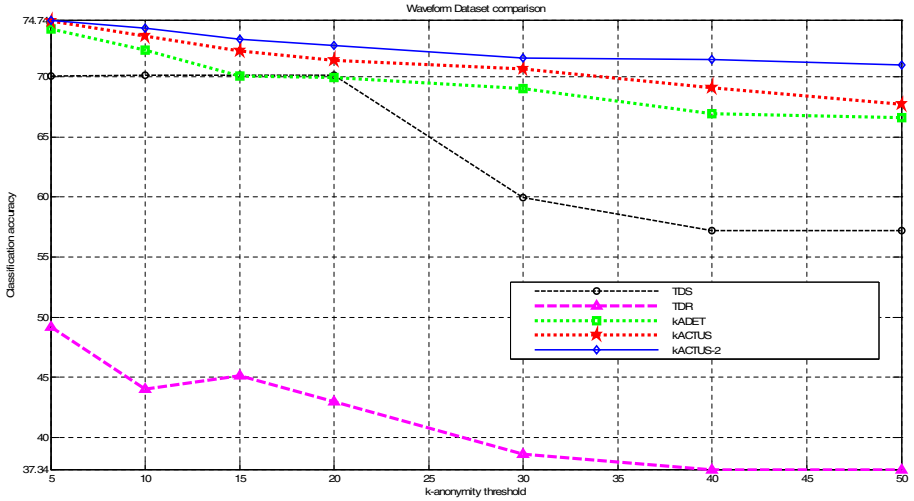

Fig. 5. Waveform. Classification accuracy vs. k-threshold. C4.5 inducer.

algorithms. Finally after $k = 25$, kACTUS-2 outperforms kACTUS. The k-thresholds are $5, 10, 15, 20, 30$. The original classification accuracy is 67.63.

Figure 5 presents classification accuracy of five algorithms for Waveform with $k = 5, 10, 15, 20, 30, 40, 50$. The original classification accuracy is 74.78. We can see that kACTUS-2 performance deteriorates very slowly and its graph is almost horizontal. kACTUS performance is slightly worse than kACTUS-2 but better than kADET, TDS and TDR. Since we used all 40 attributes of the Waveform dataset as quasi-identifiers, the performance graph can suggest that kACTUS-2,

Table 1. Area Under the Curve (AUS)

Dataset	Classification Algorithm	TDS	TDR	kADET	kACTUS	kACTUS-2
Adult 14qi/14	C4.5	76414.025	75472.2	80843.275	81830.7	**82663.22**
	PART	76401.425	75472.225	N/A	81849.25	**82338.2**
	Naïve Bayes	75295.975	73071.75	N/A	82383.8	**82755.32**
	Logistics R.	76377.775	76477.2	N/A	80761.575	**81623.65**
German Credit	C4.5	6583.175	6621.625	6596.7	6645.725	**6646.4**
	PART	6456.275	6292.65	N/A	6438.1	**6505.225**
	Naïve Bayes	6585.775	6173.25	N/A	**6640.52**	6625.95
	Logistics R.	6635.25	6525.025	N/A	**6635.35**	6626.425
TTT	C4.5	1731.175	1658.75	1771.3	1789.725	**1802.35**
	PART	1727.925	1628.5	N/A	**1795.325**	1791.15
	Naïve Bayes	**1751.525**	1716.075	N/A	1623.675	1566.675
	Logistics R.	**1738.275**	1693.15	N/A	1481.8	1389.725
Glass	C4.5	1292.3	964.475	1350.35	**1383.2**	1372.225
	PART	1292.3	964.475	N/A	**1400.675**	1390.225
	Naïve Bayes	1202.8	1289.625	N/A	1387.525	**1463.15**
	Logistics R.	**1284.65**	965.9	N/A	1188.9	1242.9
Waveform	C4.5	2860.2	1837.65	3113.85	3185.825	**3252.475**
	PART	2857.85	2175.825	N/A	3208.5	**3231.75**
	Naïve Bayes	2827.9	2064.7	N/A	2961.075	**3054.7**
	Logistics R.	**2859.825**	1035.925	N/A	2038	2208.4

kACTUS and kADET can work with a large number of quasi-identifiers while TDS and TDR cannot handle a large number of quasi-identifiers without losing classification accuracy. This is clearly seen in the example of the German dataset in Figure 2.

Because the curves of the compared algorithms might intersect, we used the Area Under Curve (AUC) measurement as a single value metric to compare algorithms and establish a dominance relationship among them. A better algorithm should have a larger area. The reported values in Table 1 represent the mean AUC performance in the Adult, German, TTT, Glass and Waveform datasets respectively. The results indicate that kACTUS-2 outperforms the other four algorithms in 11 cases out of a total of 20. kACTUS-2 outperforms TDS in 15 cases out of 20 and TDR in 18 cases out of 20. Moreover kACTUS-2 outperforms kADET using C4.5 in all datasets. Note that since the output of kADET is an anonymous C4.5 decision tree rather than an anonymous dataset, we cannot compare it to other classification algorithms. More important, kACTUS-2 outperforms kACTUS in 13 out of 20 cases. In contrast to the kACTUS algorithm, kACTUS-2 does not use a random generator thus it provides more stable results, and it achieves an average reduction of 0.5% in classification error with respect to kACTUS.

In order to check whether all five algorithms construct classifiers over a test set with the same error rate, we followed the procedure proposed in [22]. First an adjusted Friedman test was used in order to reject the null hypothesis and then a Bonferroni-Dunn test was applied to examine whether kACTUS-2 performs significantly better than the other four algorithms. We found that kACTUS-2 statistically outperforms TDS, TDR, kADET, kACTUS with a confidence level of 95%.

5.3 Information Loss

A metric which is of great importance for us is information loss. Information loss in our case is the number of missing values contained in a dataset divided by the total number of values. Suppression is not as flexible as generalization, so one would anticipate a higher information loss rate in kACTUS-2 than in TDS and TDR. However, this is not the case as can be seen in Table 2, which displays suppression rates in percentages for five datasets. Since the output of kADET is an anonymized model rather than a dataset we cannot obviously determine its rate of information loss.

The results are very encouraging. From Table 2 we can see that the kACTUS-2 information loss rate is lower in 19 cases out of a total of 32 cases. The kACTUS-2

Table 2. Percentage of missing values vs. k-threshold

Dataset	k-anonymity	TDS	TDR	kACTUS	kACTUS-2
Adult 14qi/14	5	0.73	**0.71**	0.72	**0.71**
	20	**0.63**	0.67	0.72	0.72
	50	0.69	**0.67**	0.73	0.72
	100	0.75	**0.73**	0.75	**0.73**
	500	0.83	**0.73**	0.78	0.75
	1000	0.76	**0.75**	0.83	0.76
TTT	5	0.67	0.69	0.62	**0.58**
	10	0.7	0.72	0.67	**0.61**
	15	0.7	0.75	0.73	**0.64**
	20	0.71	0.76	0.74	**0.65**
	30	0.8	0.78	0.77	**0.68**
Glass	5	0.66	0.67	0.57	**0.51**
	10	0.67	0.70	0.61	**0.55**
	15	0.72	0.71	0.64	**0.58**
	20	0.75	0.77	0.69	**0.62**
	30	0.78	0.78	0.75	**0.67**
Waveform	5	0.82	0.86	0.82	**0.81**
	10	0.85	0.88	0.83	**0.81**
	15	0.85	0.88	0.83	**0.82**
	20	0.85	0.89	0.84	**0.83**
	30	0.85	0.89	0.85	**0.83**
	40	0.85	0.9	0.85	**0.84**
	50	0.87	0.91	0.86	**0.84**
German Credit	5	**0.31**	**0.31**	0.34	0.34
	10	**0.32**	**0.32**	0.36	0.35
	15	0.35	**0.33**	0.36	0.35
	20	0.36	**0.34**	0.37	0.35
	30	0.36	**0.34**	0.37	0.37
	40	0.36	**0.34**	0.38	0.38
	50	0.36	**0.35**	0.39	0.39
	80	**0.36**	0.37	0.42	0.41
	100	**0.36**	0.38	0.42	0.42

rate is lower for all values of k in the TTT, Glass and Waveform datasets. The kACTUS-2 rate is lower in two cases out of six cases in the Adult dataset and higher in the German dataset for every value of k.

6 Conclusions

In this paper we presented a new method of using k-anonymity for preserving privacy in classification tasks. The proposed method requires no prior knowledge and can be used by any inducer. It combines *compensation by suppression* presented in [15] and *compensation by swapping* which decreases information loss induced by the suppression approach. The new method also shows a higher predictive performance and less information loss when compared to existing state-of-the-art methods.

Issues to be studied further include: examining kACTUS-2 in relation to other decision trees inducers. kACTUS-2 should also be extended to other data mining tasks (such as clustering and association rules) and anonymity measures (such as l-diversity) which respond to different known attacks against k-anonymity, such as homogeneous and background attacks.

References

1. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10(5), 557–570 (2002)
2. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression
3. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression (2002)
4. Iyengar, V.S.: Transforming data to satisfy privacy constraints. In: KDD 2002: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 279–288. ACM, New York (2002)
5. Wang, K., Yu, P.S., Chakraborty, S.: Bottom-up generalization: a data mining solution to privacy protection. In: Proc. of the 4th IEEE International Conference on Data Mining (ICDM 2004) (November 2004)
6. Fung, B.C.M., Wang, K., Yu, P.S.: Top-down specialization for information and privacy preservation. In: Proc. of the 21st IEEE International Conference on Data Engineering (ICDE 2005), Tokyo, Japan, April 2005, pp. 205–216 (2005)
7. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: efficient full-domain k-anonymity. In: SIGMOD 2005: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 49–60. ACM, New York (2005)
8. Friedman, A., Schuster, A., Wolff, R.: k-anonymous decision tree induction. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 151–162. Springer, Heidelberg (2006)
9. Fung, B.C.M., Wang, K.: Anonymizing classification data for privacy preservation. IEEE Trans. on Knowl. and Data Eng. 19(5), 711–725 (2007); Fellow-Philip S. Yu
10. Friedman, A., Wolff, R., Schuster, A.: Providing k-anonymity in data mining. VLDB J. (2008) (accepted for publication)

11. Dalenius, T., Reiss, S.P.: Data-swapping, a Technique for Disclosure Control. Program in Computer Science and Division of Engineering. Brown University (1978)
12. Reiss, S.P.: Practical data-swapping: the first steps. ACM Trans. Database Syst. 9(1), 20–37 (1984)
13. Richard, A., Moore, J.: Controlled data-swapping techniques for masking public use microdata sets. Statistical Research Division Report Series, RR96-04, U.S. Bureau of the Census (1996)
14. Fienberg, S.E., McIntyre, J.: Data swapping: Variations on a theme by dalenius and reiss. Technical report, National Institute of Statistical Sciences, Research Triangle Park, NC (2003)
15. Kisilevich, S., Elovici, Y., Shapira, B., Rokach, L.: A multi-dimensional suppression for k-anonymity (to appear, 2009)
16. Shannon, C.E.: A mathematical theory of communication. Bell Systems Technical Journal 27, 379–423 (1948)
17. Newman, C.B.D., Merz, C.: UCI repository of machine learning databases (1998)
18. Witten, I.H., Frank, E.: Data mining: practical machine learning tools and techniques with java implementations. SIGMOD Rec. 31(1), 76–77 (2002)
19. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. 10(7), 1895–1923 (1998)
20. Salzberg, S.L.: C4.5: Programs for Machine Learning by J. Ross Quinlan. Machine Learning 16(3), 235–240 (1994)
21. Cessie, S.L., Houwelingen, J.C.V.: Ridge estimators in logistic regression. Applied Statistics 41(1), 191–201 (1992)
22. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)