Author: M. Zahidul Islam and L. Brankovic

Title: Privacy Preserving Data Mining: A Noise Addition Framework Using a Novel Clustering Technique

Journal: Knowledge-Based Systems **ISSN**: 0950-7051

Year: 2011

Volume: 24

Issue: 8

Pages: 1214-1223

Abstract: During the whole process of data mining (from data collection to knowledge discovery) various sensitive data get exposed to several parties including data collectors, cleaners, preprocessors, miners and decision makers. The exposure of sensitive data can potentially lead to breach of individual privacy. Therefore, many privacy preserving techniques have been proposed recently. In this paper we present a framework that uses a few novel noise addition techniques for protecting individual privacy while maintaining ahigh data quality. We add noise to all attributes, both numerical and categorical. We present a novel technique for clustering categorical values and use it for noise addition purpose. A security analysis is also presented for measuring the security level of a data set.

Author Address: zislam@csu.edu.au
ljiljana.brankovic@newcastle.edu.au

Privacy Preserving Data Mining: A Noise Addition Framework Using a Novel Clustering Technique

Md Zahidul ISLAM[1], and Ljiljana BRANKOVIC[2]

1. School of Computing and Mathematics, Charles Sturt University, Boorooma Street, NSW 2678, Australia.
Email: zislam@csu.edu.au, Corresponding author.
2. School of Electrical Engineering and Computer Science, Newcastle University, Callaghan NSW 2308, Australia.
Email: ljiljana.brankovic@newcastle.edu.au

Abstract

During the whole process of data mining (from data collection to knowledge discovery) various sensitive data get exposed to several parties including data collectors, cleaners, preprocessors, miners and decision makers. The exposure of sensitive data can potentially lead to breach of individual privacy. Therefore, many privacy preserving techniques have been proposed recently. In this paper we present a Framework that uses a few novel noise addition techniques for protecting individual privacy while maintaining a high data quality. We add noise to all attributes, both numerical and categorical. We present a novel technique for clustering categorical values and use it for noise addition purpose. A security analysis is also presented for measuring the security level of a data set.

Keywords:
privacy preserving data mining, clustering categorical values, data perturbation.

1. Introduction

During the whole process of data mining (from collection of data to discovery of knowledge) sensitive information contained in data sets may get exposed to several parties. Disclosure of such sensitive information is considered as breach of individual privacy. Consequently, there is a huge public awareness and concern about privacy. Many recent surveys illustrate this

concern [1]. Public awareness of privacy and lack of public trust in organizations may introduce additional complexity to data collection. As a result, organizations may not be able to fully enjoy the benefits of data mining.

Therefore, many privacy preserving data mining techniques have been proposed. For example, a group of privacy preserving techniques produces synthetic data from an original data set, and instead of the original data set it releases the synthetic data set that maintains some characteristics of the original data set [2], [3], [4]. Another group of techniques adds noise to a data set in order to preserve individual privacy in the perturbed data set while allowing a data miner to produce a high quality decision tree from the perturbed data set [5], [6], [7]. Many techniques have also been proposed for privacy preservation in association rule mining [8, 9, 10, 11, 12, 13, 14, 15, 16, 17].

In this paper we present a framework for adding noise to all attributes (both numerical and categorical) in two steps; in the first step following a data swapping technique [6, 18] we add noise to sensitive class attribute values, which are also known as labels. Additionally, in the next step we add noise to all non-class attributes to prevent re-identification of a record with high certainty and disclosure of a sensitive class value. Noise addition to non-class attributes also protect the attributes from being disclosed. The main goal of our noise addition techniques is to provide high level of security while preserving a good data quality.

The organization of the paper is as follows. Section 2 presents *DETECTIVE*, a novel technique for clustering categorical values. In Section 3, a few novel techniques are presented for noise addition to numerical and categorical attributes of a data set. Section 4 presents our Framework that combines the noise addition techniques in order to perturb all attributes of a data set. We also present an Extended Framework that incorporates an existing technique called *GADP* [3] or one of its variants such as *CGADP* [19] or *EGADP* [20]. Experimental results are presented in Section 5. In Section 6 we present a security analysis. Section 7 gives concluding remarks.

2. *DETECTIVE*: A Novel Categorical Values Clustering Technique

Various relationships between different classifier (non-class) attributes and the class attribute of a data set are discovered in the form of logic rules (patterns) by a decision tree. Figure 1 is an example of a decision tree.

The tree has four leaves. Leaf 1, Leaf 2 and Leaf 4 are heterogenous leaves since the records belonging to each of the leaves have different class values. Records belonging to a heterogenous leaf have different class values. We refer to the path from the root to a leaf as the *leaf-path* for that particular leaf. We call the attributes tested at the nodes of a *leaf-path* as *Leaf Influential Attributes (LINFAs)* of the leaf. Remaining attributes of the data set are called *Leaf Innocent Attributes (LINNAs)* of the leaf. Two leaves are called siblings if their leaf-paths differ only in the last node. For example, Leaf 3 and Leaf 4 of Figure 1 are sibling leaves.

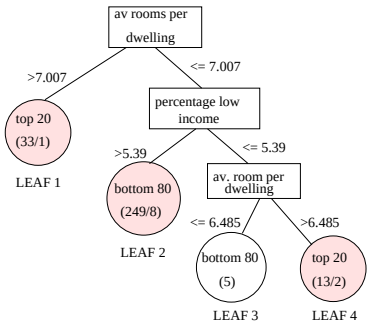

Figure 1: An example of a decision tree classifier.

We present a novel technique called *DEcision TreE based CaTegorIcal ValuE clustering technique (DETECTIVE)* [21]. In order to cluster a categorical attribute, say "A", *DETECTIVE* first builds a decision tree (using an existing algorithm such as See4.5 [22] and EXPLORE [23]) that considers A as the class attribute. Suppose L_h is a heterogeneous leaf having the majority class C_p (i.e. the class value of the majority number of records is C_p), and the minority class C_q. *DETECTIVE* considers the values C_p and C_q of attribute A as a cluster for the records belonging to L_h. The strength of their similarity can be calculated by multiplying $|R_{cp}|$ (the number of records having C_p) and $|R_{cq}|$ (the number of records having C_q) in R_h. Additionally, let the leaves L_i and L_{i+1} be two sibling leaves having class values C_p and C_r, respectively. *DETECTIVE* considers C_p and C_r as a cluster within the horizontal segment containing the set of records $R_u = R_i \cup R_{i+1}$. Again their strength can be calculated by multiplying $|R_{cp}|$ and $|R_{cr}|$ in R_u. Note that similarity of values in a heterogeneous leaf are considered stronger than the similarity of majority class values belonging to sibling leaves.

2.1. EX-DETECTIVE

EX-DETECTIVE is an extension of *DETECTIVE* for producing a cluster of records. In *EX-DETECTIVE* we consider the records belonging to a leaf as similar with respect to the attribute A that has been considered as the class attribute for the tree. Each leaf (homogeneous or heterogeneous) produces a set of records that are similar to other records of the set. Similarly, in another tree that considers attribute B as the class attribute, the records belonging to a leaf are considered to be similar to each other with respect to B. There can be a set of common records that are similar with respect to both A and B if they are found similar to each other in both trees. These common records are considered to be a cluster of records with respect to A and B. A user can select a set of attributes that he/she wants to use for producing the clusters. Moreover, the user can put different importance (or weights) on different chosen attributes by pruning corresponding trees differently. The less weight we assign the more we prune. The maximum pruning producing only one leaf that contains all records of the data set, will be the result of zero weight assigned.

If a data set contains both numerical and categorical attributes *EX-DETECTIVE* first obtains clusters of records based on chosen categorical attributes. Within each cluster it then applies a conventional clustering technique such as distance-based and k-means clustering on the numerical attributes only. Finally, *EX-DETECTIVE* produces a number of clusters where all records belonging to a cluster are similar to each other with respect to the chosen categorical and numerical attributes.

3. Our Noise Addition Techniques

*3.1. **Class Attribute Perturbation Techniques***

We use the following notations to explain our class attribute perturbation techniques [24, 25]. Let, H be the number of heterogeneous leaves, m_k be the number of majority records in the kth heterogeneous leaf where $1 \leq k \leq H$, n_k be the number of minority records (i.e. the records not having the class values same as the majority class value) in the kth heterogeneous leaf where $1 \leq k \leq H$, and $E(N)$ be the expected number of changed class values.

In *Random Perturbation Technique (RPT)*, the class values of all minority records n_k belonging to the kth heterogeneous leaf are first converted from the minority class to majority class. Then n_k records are randomly selected from the set of records belonging to the leaf and their class values are changed

to a minority class proportionate to the number of original records belonging to the class. This change is made in all heterogeneous leaves.

The expected number of changed classes in the whole perturbed data set (for all heterogeneous leaves) is,

$$E(N) = \sum_{k=1}^{H} \frac{2m_k n_k}{m_k + n_k}. \quad (1)$$

In *All Leaves Probabilistic Technique (ALPT)* we perturb all the records of the data set, instead of just the records within heterogeneous leaves. We use this method as a simulation of a natural noise occurring in the class attribute. We compare RPT technique with this one in order to evaluate the effectiveness of RPT in pattern preservation. In *ALPT*, we change the class of all records in the data set with the probability $p = \frac{1}{N_{Total}} \sum_{k=1}^{H} \frac{2m_k n_k}{m_k+n_k}$, where N_{Total} is the total number of records in the data set. The expected number of changed classes in the whole perturbed data set is the same as the one in RPT.

3.2. ***Non-class Numerical Attribute Perturbation Techniques***

In order to add noise to both *LINNAs* and *LINFAs* we present two novel noise addition techniques called *Leaf Innocent Attribute Perturbation Technique (LINNAPT)* and *Leaf Influential Attribute Perturbation Technique (LINFAPT)* [26, 24, 25]. The effectiveness of these techniques in maintaining the data quality is evaluated through comparing them with another technique called *Random Noise Addition Technique (RNAT)*, which is not catered for preserving the patterns unlike the other techniques.

3.2.1. ***The Leaf Innocent Attribute Perturbation Technique (LINNAPT)***

We first build a decision tree from an unperturbed data set. The set of numerical *LINNAs* and the set of numerical *LINFAs* are then determined for the records belonging to a leaf. Let A be a numerical *LINNA*. *LINNAPT* adds noise to A and produces a perturbed attribute $A^* = A + \xi$, where ξ is a discrete noise with a mean μ and a variance σ^2. *LINNAPT* keeps the domain of A^* the same as the domain of A. Therefore, it takes a wrap around approach, if A^* falls outside the domain of A.

3.2.2. ***The Leaf Influential Attribute Perturbation Technique (LINFAPT)***

LINFAPT adds noise to B (a numerical *LINFA*) and produces a perturbed attribute $B^* = B + \xi$, where ξ is a noise generated from a normal distribution with a mean μ and a variance σ^2. The main difference of *LINFAPT* with *LINNAPT* is that in *LINFAPT* the range of a *LINFA* (for a leaf) is defined by the conditional values of the *LINFA*. For example, in Figure 1 "av rooms per dwelling" is a *LINFA* for both Leaf 1 and Leaf 4. The ranges of the *LINFA* for the two leaves are > 7.007 and within [6.486, 7.007], respectively. *LINFAPT* also uses a wrap around approach when a perturbed value B^* falls outside the range. The distribution of the noise can be chosen in both *LINNAPT* and *LINFAPT* to suit a particular application. Moreover, they add noise to each and every record of a data set.

3.2.3. ***The Random Noise Addition Technique (RNAT)***

Random Noise Addition Technique (RNAT) adds noise having zero mean and a uniform distribution. For example, if the domain size of an attribute is $|D|$, *RNAT* generates a pseudo-random number n from an uniform distribution having a range between $-(D-1)$ and $+(D-1)$. The random number n is then added with an attribute value x to produce the perturbed value $p = x + n$. The domain of an attribute is maintained through a wrap around approach when a perturbed value falls outside the domain.

3.3. ***The Categorical Attribute Perturbation Technique (CAPT)***

Categorical Attribute Perturbation Technique (CAPT) first uses DETECTIVE to cluster categorical values and then within each cluster it changes a categorical value (with a predefined probability) to another categorical value belonging to the same cluster.

We first introduce a few terms as follows, before we present the steps of CAPT in detail.

u - A user defined probability. It is the probability of making any changes to a categorical attribute in the first place.

p - Another user defined probability. It is the probability to change a class value of a leaf, to the majority class of its sibling leaf. If there are t number of sibling leaves then p_i will be the probability of changing the class value to the majority class of the ith sibling leaf, where $1 \leq i \leq t$. In that case, $p = \sum_{i=1}^{t} p_i$, where $p_i = p_j; \forall i \neq j$

$(1-p)$ - This is the probability of shuffling the class values within a heterogeneous leaf.

q - probability of assigning the class value of a record belonging to a leaf, as the majority class of the leaf. $q = \frac{m}{m+\sum_{i=1}^{k} n_i}$, where m is the number of records having the majority class of the leaf, n_i is the number of records having the ith minority class of the leaf, and k is the number of minority classes in the leaf.

l_i - probability of assigning the class value of a record belonging to a leaf to the ith minority class of the leaf. $l_i = \frac{n_i}{m+\sum_{j=1}^{k} n_j}$.

We now present a high level pseudo-code of CAPT for adding noise to a categorical attribute A of a data set. CAPT first builds a decision tree T_A that considers attribute A as the class attribute. It then takes user input for u and p.

For each record, DO:

Step 1: Determine the leaf L_h (of T_A) that the record belongs to. Also determine all sibling leaves (if any) of L_h.

Step 2: Change the value of A to the majority class of the ith sibling leaf of L_h with a probability $u * p_i$. The probability of changing the value of A to the majority class of any sibling leaf of L_h will be $u * p$. If there is no sibling leaf do nothing for the step.

Step 3: If the value of A is not changed in Step 2, change the value to the majority class of L_h with a probability $u * (1-p) * q$.

Step 4: If the value of A has not been changed in Step 2 or Step 3, change the value to the ith minority class of L_h with a probability $u * (1-p) * l_i$.

END DO

3.4. ***Random Categorical Technique***

Random Categorical Technique changes a categorical value, by a user defined probability u, to any other value belonging to the same attribute. Let, A be a categorical attribute with domain $A = \{a_1, a_2, \ldots a_n\}$ having the domain size $|A| = n$. If a record has the attribute value a_i, where $1 \leq i \leq n$ then Random Categorical technique will change the value to a_j, where $1 \leq j \leq n$ and $i \neq j$, by the probability $u \times \frac{1}{n-1}$. This technique is mainly used in this study to evaluate the effectiveness of CAPT.

4. The Framework and The Extended Framework

4.1. ***The Framework***

We now present a high level pseudocode of our *Framework* [26] for adding noise to all attributes. A decision tree is first built from an original data set, and then used to perturb the data set as follows.

For each leaf, DO:

Step 1: Add noise to numerical *Leaf Influential Attributes (LINFAs)* of the original records belonging to the leaf by *Leaf Influential Attribute Perturbation Technique (LINFAPT)*. Thereby produce a set of perturbed records p_{s1}.

Step 2: Add noise to numerical *Leaf Innocent Attributes (LINNAs)* of the original records belonging to the leaf by *Leaf Innocent Attribute Perturbation Technique (LINNAPT)*. Thus produce another set of perturbed records p_{s2}.

Step 3: Add noise to the categorical attributes of the original records belonging to the leaf by *CAPT*. Thereby produce another set of perturbed records p_{s3}.

Step 4: If the leaf is heterogeneous add noise to the class attribute, of the original records belonging to the leaf, by *Random Perturbation Technique (RPT)*. In this way another set of perturbed records p_{s4} is produced.

Step 5: Produce a set of combined perturbed records p_c, where $|p_c| = |p_{si}|, 1 \leq i \leq 4$, in such a way so that only the perturbed attributes from each p_{si} are included into p_c.

END DO

Each of the steps from Step 1 to Step 4 takes a set of unperturbed records and produces a set of perturbed records where values belonging to one or more attributes are perturbed. For example, Step 1 produces a set of perturbed records p_{s1} where the numerical *LINNAs* are perturbed. If a data set does not have any non-class numerical attributes or if a leaf does not have any numerical *LINNAs* then Step 1 produces a set of records which is the same as the original set of records. The *Framework* repeats the process for all leaves and thereby perturbs the whole data set.

4.2. ***The Extended Framework***

We now present an extension of our *Framework* in order to preserve the original statistical parameters in addition to the data mining patterns such as the logic rules. Our *Extended Framework* has the following steps [26].

For each leaf, DO:

Step 1: We consider the collection of records belonging to the leaf as a data set in its own right. We add noise to the records belonging to the leaf using either *GADP*, or *CGADP* or *EGADP* technique [3, 19, 20], where the domain of each numerical *LINFA* is bounded by the range defined by the conditional values of the *LINFA* in the decision tree. A set of perturbed records p_{s1} is thereby produced.

Step 2: Add noise to the categorical attributes, of the original records belonging to the leaf, by *CAPT*. Thereby produce a set of perturbed records p_{s2}.

Step 3: If the leaf is heterogeneous - add noise to the class attribute, of the original records belonging to the leaf, by *Random Perturbation Technique (RPT)*. Thus produce a set of perturbed records p_{s3}.

Step 4: Produce the combined perturbed records containing all perturbed attributes from p_{s1}, p_{s2} and p_{s3}.

END DO

5. Experimental Results

In this section we first present experimental results on DETECTIVE and CAPT on a synthetic data set. We then present experimental results on our Framework and Extended Framework using two real data sets. We compare our frameworks with GADP [3] and random noise addition approaches.

We now introduce a few terms that we use throughout this section. The decision tree obtained from an original training data set is called original tree. The rules belonging to the original tree are called original rules. Similarly, the decision tree obtained from a perturbed data set and the rules belonging to the tree are called perturbed tree and perturbed rules, respectively. The original rule which is the most similar to a perturbed rule is denoted as the best matching original rule of the particular perturbed rule. Finally, the weight of a rule is the number of records which satisfy the rule.

Based on the degree of similarity of a perturbed rule and the best matching original rule, the perturbed rules are informally categorized in four different types, Type A, Type B, Type C and Type D. We next give an example of a rule set that we shall subsequently use to illustrate Type A, Type B, Type C and Type D.

Let **P**, **Q**, **R** and **S** be non-class attributes, and let **C** be the class attribute of a data set. Suppose **P**, **Q** and **R** are numerical attributes with domains [1,10]. We assume **S** and **C** are categorical attributes with domain

$\{s_1, s_2, s_3\}$ and $\{c_1, c_2\}$, respectively. Suppose there are altogether five original rules $R_o(1)$, $R_o(2)$, $R_o(3)$, $R_o(4)$ and $R_o(5)$ as follows.

$R_o(1) : \mathbf{P}(> 5)$ *and* $\mathbf{Q}(\leq 2)$ *and* $\mathbf{S}(= s_1) \Rightarrow c_1$,
$R_o(2) : \mathbf{P}(> 5)$ *and* $\mathbf{Q}(\leq 2)$ *and* $\mathbf{S}(= s_2) \Rightarrow c_2$,
$R_o(3) : \mathbf{P}(> 5)$ *and* $\mathbf{Q}(\leq 2)$ *and* $\mathbf{S}(= s_3) \Rightarrow c_1$,
$R_o(4) : \mathbf{P}(> 5)$ *and* $\mathbf{Q}(> 2) \Rightarrow c_2$, and
$R_o(5) : \mathbf{P}(\leq 5) \Rightarrow c_2$.

- **Type A**: A "Type A" rule is exactly the same as its best matching original rule. For example, a perturbed rule $R_p(1)$: $\mathbf{P}(> 5)$ *and* $\mathbf{Q}(\leq 2)$ *and* $\mathbf{S}(= s_1) \Rightarrow c_1$, is identical to its best matching original rule $R_o(1)$ and therefore, $R_p(1)$ is defined as Type A.

- **Type B**: A perturbed rule can be considered as Type B if the perturbed rule differs from its best matching original rule only in conditional values of corresponding numerical attribute/s. We also consider a set of rules as Type B if 2 or more rules that differ in only one attribute can be combined into a rule that differs from an original rule only in combined values. For example, a perturbed rule $R_p(2)$: $\mathbf{P}(> 6)$ *and* $\mathbf{Q}(\leq 2)$ *and* $\mathbf{S}(= s_1) \Rightarrow c_1$, differs from its best matching original rule $R_o(1)$ only in the conditional value for attribute $\mathbf{P}$. Therefore, $R_p(2)$ is defined as Type B. Let us give another example of a Type B rule. Let $R_p(21)$: $\mathbf{P}(> 6)$ *and* $\mathbf{Q}(> 4)$ *and* $\mathbf{S}(= s_1, s_2) \Rightarrow c_2$ and $R_p(22)$: $\mathbf{P}(> 6)$ *and* $\mathbf{Q}(> 4)$ *and* $\mathbf{S}(= s_3) \Rightarrow c_2$ then combining $R_p(21)$ and $R_p(22)$ we get $\mathbf{P}(> 6)$ *and* $\mathbf{Q}(> 4) \Rightarrow c_2$ which is a Type B rule since it differs from $R_o(4)$ only in conditional values for attribute P and Q.

- **Type C**: Type C is a rule that does not involve any Total Innocent Attribute of an original tree and is not of Type A or Type B. If an attribute is a Leaf Innocent Attribute for each leaf then we call it Total Innocent Attribute.

- **Type D**: If a perturbed rule involves at least one Total Innocent Attribute then the rule is defined as Type D. For example, a perturbed rule $R_p(3)$: $\mathbf{P}(> 8)$ *and* $\mathbf{Q}(\leq 2)$ *and* $\mathbf{R}(< 3) \Rightarrow c_1$ involves Total Innocent Attribute R and therefore, we denote $R_p(3)$ as Type D.

We measure the similarity of a perturbed tree $T_p(1)$ with the original tree T_o by evaluating the types of the perturbed rules and their corresponding weights. For example, consider that a perturbed tree $T_p(1)$ has "Type A" rules which apply to 90% of records and "Type D" rules that apply to the 5% of the records belonging to the perturbed data set. Also assume that another perturbed tree $T_p(2)$ has "Type D" rules that apply to 70% of the records and "Type A" rules that apply to 5% of the records belonging to the corresponding perturbed data set. Therefore, $T_p(1)$ and the original tree are considered more similar than the $T_p(2)$ and the original tree.

5.1. ***Experiments on DETECTIVE***

We use 399 records of a synthetically created data set called *Customer Status (CS)* data set. *CS* data set has five categorical non-class attributes and a categorical class attribute. We apply *DETECTIVE* on the *CS* data set in order to cluster values belonging to the attribute *Car Make*. *DETECTIVE* builds a decision tree (using See5 algorithm), shown in Figure 2, that considers *Car Make* as the class attribute. It discovers similarity among Ford, Toyota and Nissan within the records belonging to Leaf 1 of the tree. It also finds a similarity between Toyota and Holden among the records belonging to Leaf 3.

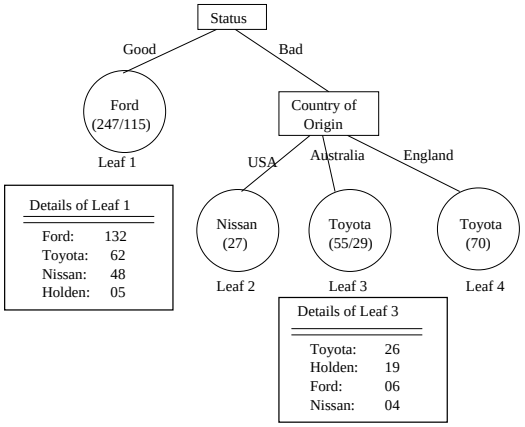

Figure 2: Details of a decision tree built from the unperturbed *CS* data set. The tree considers attribute *Car Make* as class attribute. This tree is used for clustering values of the attribute *Car Make*.

We then apply *CACTUS* [27] on the *CS* data set. According to *CACTUS* Toyota and Ford are found similar over the whole data set. DETECTIVE

performs better than *CACTUS* as it extracts various similarities within different segments of the data set.

5.2. ***Experiments on CAPT***

We apply *CAPT* on the *CS* data set and perturb the categorical non-class attribute *Car Make* with $u = 1$ and $p = 0.1$ (see Section 3.3). We run this experiment ten times and thereby produce ten perturbed data sets and ten perturbed trees. Each perturbed decision tree is extremely similar to the original decision tree. All ten perturbed trees have Type A logic rules covering 100% records.

5.3. ***Experiments on the Framework and Extended Framework***

In this set of experiments we apply our Framework and Extended Framework on two data sets; *Adult* and *Wisconsin Breast Cancer (WBC)* [28]. We perturb a data set ($DS_{training}$) by our *Framework*, where all non-class numerical attributes are perturbed by *LINFAPT* and *LINNAPT* using $\mu = 0$ and $\sigma = 33.33\%$. We note that since WBC data set does not have any non-class categorical attributes, we do not need to use CAPT for WBC data set. However, we use CAPT with u = 0.1 and p = 0 for two categorical attributes *workclass* and *marital-status* of Adult data set. For other categorical attributes we use u= 0. Moreover, the class attribute of a data set is perturbed by *RPT*. We run the Framework experiment 5 times on a data set and thereby produce five perturbed data sets $DS_{Frmwrk}i$ and five perturbed decision trees $DT_{Frmwrk}i$, where $1 \leq i \leq 5$. Similarly, we apply Extended Framework on a data set $DS_{training}$ 5 times and thereby build 5 perturbed decision trees $DT_{ExtndFrm}i$, where $1 \leq i \leq 5$. Extended Framework uses GADP for numerical attributes, CAPT for categorical attributes and RPT for the class attribute. In CAPT, we again use u = 0.1 and p = 0 for two categorical attributes *workclass* and *marital-status* and $u = 0$ for all other attributes.

In order to evaluate Framework and Extended Framework we introduce two other techniques called *Random Framework (RF)* and *Random Extended Framework (REF)*. We apply both *Random Framework* and *Random Extended Framework* on a data set five times and thereby produce five perturbed trees $DT_{Random}i$ and $DT_{RandomExt}i$, where $1 \leq i \leq 5$, respectively.

RNAT, *Random Categorical* and *ALPT* (see Section 3) are combined to form *Random Framework (RF)*. Whereas, in *Random Extended Framework (REF)* we use *GADP*, *Random Categorical* and *ALPT*. In REF, we apply

GADP on all records of a data set at a time, unlike *Extended Framework* that applies GADP separately on the records belonging to a leaf. Being consistent with all experiments, we again use $u = 0.1$ for *workclass* and *marital-status* and $u = 0$ for all other categorical attributes in *Random Categorical* technique used for both *RF* and *REF*. The "uniform distribution" of *RNAT* is calculated in such a way so that the total amount of noise in RNAT and LINNAPT remains the same.

5.3.1. Experiments on the Adult Data Set

There are 2,399 records having one or more missing values. We first remove the records with missing values and produce a data set DS_{adult} having 30,162 records.

We then divide DS_{adult} into two data sets; a training data set $DS_{training}$ having 25,600 records and a testing data set $DS_{testing}$ having 4,562 records. $DS_{testing}$ is created by taking 15% of records belonging to each leaf of DT_{adult} obtained from DS_{adult}. A decision tree $DT_{training}$ is built from $DS_{training}$. We use *See 5* decision tree builder with the default *Pruning CF* = 25% and *Minimum Records per Leaf* = 200.

Technique	Number of DTs	Type A and/or Type B Rules	Type C Rules	Type D Rules
Framework	4	100%	0%	0%
	1	70%	30%	0%
Random Framework	5	0%	0%	100%
Extended Framework	3	75% (around)	25% (around)	0%
	2	20% (around)	80% (around)	0%
Random Extended Framework	5	0%	0%	100%

Table 1: Decision tree similarity based data quality analysis of Adult data sets perturbed by various techniques.

We apply each of the four frameworks on $DS_{training}$ five times and produce five perturbed data sets for each framework. Decision trees are then built

from the perturbed data sets. We carefully analyse similarity of various perturbed trees with the original tree $DT_{training}$ as shown in Table 1. It is clear from the table that Framework best preserves the similarity of perturbed trees with the original tree.

Classifier Name	Applied On		
	A perturbed 25,600 rec.	B original 25,600 rec.	C testing 4,562 rec.
$DT_{training}$		85%	83.8%
DT_{Frmwrk}	84.8%	84.8%	83.45%
DT_{Random}	62.25%	81.3%	80.5%
$DT_{ExtndFrm}$	84.5%	82.4%	81.6%
$DT_{RandomExt}$	62.65%	79.35%	78.65%

Table 2: Prediction accuracy of the classifiers obtained from the unperturbed and various perturbed Adult data sets.

We evaluate prediction accuracy of a perturbed tree on three data sets; its underlying perturbed data set, the original data set and the testing data set. In Table 2 we present the average accuracy of the decision trees obtained from two data sets perturbed by a framework. For example, DT_{Frmwrk} presents the average accuracy of the trees obtained from two data sets perturbed by the Framework. The prediction accuracy of DT_{Frmwrk} is clearly better than the accuracy of other perturbed trees. DT_{Random} and $DT_{RandomExt}$ trees have very low accuracy on underlying perturbed data sets (where the trees have been built from) indicating that the trees do not represent their underlying perturbed data sets well. Therefore, their reasonably good accuracies on original and testing data sets fail to establish a high similarity of a perturbed data set with the original data set.

We also compare the correlation matrices of the original data set and the perturbed data sets. We first produce a mean vector MV = [38.44, 189669.70, 10.12, 1102.66, 87.39, 40.88] and a correlation matrix M for the numerical attributes of the original training data set $DS_{training}$. A correlation matrix M^x_{Frmwrk} and mean vector MV^x_{Frmwrk} are then produced from the xth perturbed data set obtained from the experiments on Framework, where $1 \leq x \leq 5$. We calculate a matrix of average absolute differences, $AvgDifMat_{Frmwrk} = \frac{\sum_{x=0}^{5} |M - M^x_{Frmwrk}|}{5}$. An average value $AvgDifValue_{Frmwrk}$

is then calculated from $AvgDifMat_{Frmwrk}$ by summing up all elements and dividing the sum by the total number of elements. We also calculate a mean vector of average absolute differences, $AvgMV_{Frmwrk} = \frac{\sum_{x=0}^{5} |MV - MV^{x}_{Frmwrk}|}{5}$. Similarly, we also calculate these parameters for Random Framework, Extended Framework, and Random Extended Framework. We now present them as follows.
$AvgDifValue_{Frmwrk} = 0.065667$.
$AvgMV_{Frmwrk}$ = [13.812, 526438.39, 0.578, 4519.55, 893.16, 10.902].
$AvgDifValue_{Random} = 0.05423$.
$AvgMV_{Random}$ = [16.8, 558775, 1.718, 48998.1, 2088.02, 9.238].
$AvgDifValue_{ExtFrm} = 0.026322$.
$AvgMV_{ExtFrm}$ = [3.212, 4556.746, 0.418, 426.75, 59.304, 0.44].
$AvgDifValue_{RandomExt} = 0.01332$.
$AvgMV_{RandomExt}$ = [2.822, 1833.68, 0.488, 2365.34, 120.828, 0.454].

5.3.2. Experiments on Wisconsin Breast Cancer (WBC) Data Set

We run exactly same set of experiments on WBC data set as the experiments on Adult data set. Additionally we apply just GADP on WBC data set and produce five perturbed data sets. Similarities of various perturbed trees with the original tree is presented in Table 3. In terms of prediction accuracy (see Table 4) Extended Framework and Framework performs better than all others, specially on the testing data set.

Mean vector of the training data set is MV = [4.44, 3.18, 3.24, 2.88, 3.23, 3.6, 3.41, 2.89, 1.63]. Average values of the average difference matrices and the average difference of mean vectors for various techniques are presented as follows.
$AvgDifValue_{Frmwrk} = 0.542815$.
$AvgMV_{Frmwrk}$ = [0.506, .060, 1.576, 1.736, 1.150, 0.21, 1.234, 1.752, 2.424].
$AvgDifValue_{Random} = 0.655802$.
$AvgMV_{Random}$ = [1.044, 2.424, 2.230, 2.614, 2.320, 1.888, 2.112, 2.704, 3.950].
$AvgDifValue_{ExtFrm} = 0.155654$.
$AvgMV_{ExtFrm}$ = [1.332, 0.24, 0.264, 0.174, 0.49, 0.268, 0.32, 0.244, 0.056].
$AvgDifValue_{RandomExt} = 0.174716$.
$AvgMV_{RandomExt}$ = [1.11, 0.726, 0.676, 0.456, 0.466, 0.714, 0.552, 0.492, 0.13].

Technique	Number of DTs	Type A and/or Type B Rules	Type C Rules	Type D Rules
Framework	4	$\geq 90\%$	$\leq 10\%$	0%
	1	$\geq 90\%$	$\leq 7.4\%$	2.67%
Random Framework	2	0%	0%	100%
	1	0%	100%	0%
	2	0%	$\leq 20\%$	$\geq 80\%$
Extended Framework	4	0%	100%	0%
	1	26%	74%	0%
Random Extended Framework	3	0%	$\leq$90%	$\geq$10%
	2	0%	$\geq$90%	$\leq$10%
GADP	4	0%	$\leq$90%	$\geq$10%
	1	0%	$\geq$90%	$\leq$10%

Table 3: Decision tree similarity based data quality analysis of WBC data sets perturbed by various techniques.

6. Security Analysis

Due to the varying definitions of a disclosure it is not trivial to measure a disclosure risk. Moreover, a disclosure risk depends on various other factors such as supplementary knowledge of an intruder and the approach taken by an intruder. However, effective measuring of disclosure risk is important as the effectiveness of a data perturbation technique is evaluated by the disclosure risk and the data quality of a perturbed data set.

Before we introduce our approach to measuring the disclosure risk, we first need to remind the reader that the main aim of our noise addition technique is twofold:

1. to prevent a disclosure of confidential individual class values contained in the data set; which we achieve in this study by introducing perturbation to all attributes, in order to make re-identification of the records difficult and in most instances even impossible.
2. to preserve the patterns discovered by the decision tree builder prior to perturbing the data set. We also note that there has been a research effort published in the literature where the goal has been to hide confidential patterns [29, 30], but that is beyond the scope of our study.

Classifier Name	Applied On		
	A perturbed 600 records	B original 600 records	C testing 83 records
$DT_{training}$		99.3%	89.2%
DT_{Frmwrk}	99.05%	98.5%	87.4%
DT_{Random}	69.6%	66.9%	65.1%
$DT_{ExtndFrm}$	98.9%	95.35%	92.2%
$DT_{RandomExt}$	87.5%	91.45%	82.5%
DT_{GADP} (without ALPT)	87.5%	91.45%	82.5%

Table 4: Prediction accuracy of the classifiers obtained from the unperturbed and various perturbed WBC data sets.

We are now ready to measure the risk of a confidential class value being disclosed to an intruder who has a full access to the perturbed data set. We assume that the intruder knows something about the record whose confidential class he/she intends to learn. That might be all of the non-class attributes, some of the attributes or perhaps just one or two of them. We call the knowledge as the supplementary knowledge of an intruder. The intruder basically has two options as follows.

- They can construct a decision tree from the perturbed data set. Then they can run the record of interest (based on his/her supplementary knowledge) through the constructed decision tree and relatively accurately estimate the class of the record.
- Alternatively, the intruder can try to re-identify the record, that is, to match the record he/she has interest in to the records of the perturbed data set, identify the best match and adopt the class of the best match as the likely class of the original record.

We next discuss each of these two options.

1. **Running the Record Through the Decision Tree:** For intruder to successfully run the decision tree and estimate the class, he/she needs to know influential attributes for the leaf the record would naturally belong to. We argue that if the intruder has such knowledge about the

record, then he/she can always learn the class, regardless whether the record is contained in the original training set or not. Thus, a record from the training set that intruder has substantial knowledge of is not under greater risk of disclosure than any other record known to the intruder. The only way to prevent this kind of disclosure would be to mask (hide) patterns. However, that would go against the main aim of our noise addition technique, which is to preserve the patterns while hiding the individual confidential values. Thus we only need to consider the option 2, which is based on re-identification of the records. We need to argue that the intruder is not likely to learn the confidential class through re-identification of the record with more certainty than he/she can learn by running the record through the decision tree. Indeed, we argue that since the decision tree obtained from a perturbed data set is very similar to the tree obtained from the original data set, the prediction accuracy of the perturbed tree is very high. We shall next discuss the accuracy of predicting the class through record re-identification.

2. **Record Re-identification:** First of all, we note that it is not always a trivial job to successfully re-identify a target record for the following reasons. First, an intruder needs enough supplementary knowledge about the individual and about the perturbation technique. Often an intruder knows something about the target record, but may not know precise and sufficient information to match the attribute values that appear in a released data set. For example, an intruder may know that the salary of a target record is between 50K and 60K, but may not know the exact salary. In general, the greater the intruder's supplementary knowledge the higher the risk of disclosure. Therefore, in order to perform a conservative disclosure risk assessment we assume that an intruder knows every non-class attribute value of the target record. We also assume that the intruder knows our data perturbation technique and the distribution of noise. Additionally, the intruder knows that the target record exists in the released data set.
   A disclosure risk depends on various factors such as the technique used by an intruder for learning a sensitive value and the supplementary knowledge of an intruder. We assume that an original data set having n records and m attributes is perturbed by a technique such as the *CAPT*, *LINFAPT* and *LINNAPT*, and thereby a perturbed data set having n records and m attributes is produced. We present *An Entropy Based*

Technique for measuring disclosure risk in two different ways; 1. a risk of re-identification where the intruder can re-identify the record and learn all the attributes with a certain probability, 2. a risk of intruder learns the class of the target record. Note that it may be possible for the intruder precisely learn the class even when re-identification is not possible. This will happen when all or most suspect perturbed records have the same class.

We first consider re-identification, that is, a scenario where an intruder takes a probabilistic approach to identify the target record in a perturbed data set. Let us assume that his/her target record is the *xth* record of an original data set. Recall that the intruder has an access to a released data set and does not have access to the original data set. For each record i of the perturbed data set the intruder can calculate the probability that a record has been perturbed from the target record as follows.

$$P_{xi} = \frac{\prod_{j=1}^{m} P_{xi}^{j}}{\sum_{i=1}^{n}(\prod_{j=1}^{m} P_{xi}^{j})} \quad (2)$$

where, P_{xi} is the probability that the *ith* record of the perturbed data set corresponds to the *xth* record of the original data set, i.e., the target record;

P_{xi}^{j} is the probability that the *jth* attribute value of the *ith* record of the perturbed data set is the perturbed value of the *jth* attribute value of the *xth* record of the original data set;

m is the total number of non-class attributes; and

n is the total number of records.

The main challenge in the above formula is to calculate P_{xi}^{j}. We assume that an intruder has some supplementary knowledge about the target record (i.e. the record of interest) and knows our noise addition technique. However, since an intruder does not have access to the original data set he/she does not know the original tree and the leaf which the target record x belongs to.

6.1. P_{xi}^{j} Calculation for Categorical Attributes

In order to estimate P_{xi}^{j} for a categorical attribute J in a target record, the best the intruder can do is to assume that the perturbed decision tree $T_p(J)$ (that considers the *jth* attribute as the class) is the same as the original decision tree $T_o(J)$. Based on the assumption he/she can run the target

record (using his supplementary knowledge) through $T_p(J)$ and learn the leaf L that the target record arrives to. From this he/she can learn the majority classes of the sibling leaves of L and also adopt the distribution of the class attribute values of L. The intruder can calculate P^j_{xi} from the information.

For example, let us first consider a scenario where L does not have any sibling leaf. Let the distribution of class values of L be n_1, n_2, $\cdots$ n_k number of records for the class values J_1, J_2 $\cdots$ J_k, respectively where J=$\{J_1, J_2, \cdots J_k\}$ and k is the domain size of J. If in the *ith* perturbed record, value of the attribute J is J_l then an intruder can calculate

$$P^j_{xi} = \frac{n_l}{\sum_{i=1}^{k} n_i} \times u \times (1-p), \tag{3}$$

where p is the user defined probability of changing a class value to the majority class of the sibling leaf/leaves as introduced in CAPT in Section 3.3, and u is a user defined probability of changing a categorical value as introduced in Section 3.3.

Now, let us consider another scenario where L has S number of sibling leaves. In this case the intruder can calculate

$$P^j_{xi} = \{\frac{n_l}{\sum_{i=1}^{k} n_i} \times (1-p) + \sum_{s=1}^{S}(\frac{p}{S} \times P_s)\} \times u \tag{4}$$

where,

S is the number of sibling leaves of L.

S_s is the *s-th* sibling leaf.

$P_s = 1$, if the majority class of S_s is J_l, otherwise $P_s = 0$.

6.2. P^j_{xi} Calculation for Numerical Attributes

For a numerical attribute an intruder needs to take a slightly different approach. Let us assume that from his supplementary knowledge an intruder knows the actual value belonging to an attribute in a target record. He/she can calculate the probability P^j_{xi} from the distribution of noise. For example, if we have the *jth* attribute value equal to 5 in both original and a perturbed data set then we estimate P^j_{xi} is 40% given the probability of zero noise is 40%. However, in our Framework the distribution of noise actually depends on the leaf of the original decision tree that contains the target record x. To illustrate this point, let us consider a decision tree obtained from original

WBC data set. Each non-class attribute of WBC data set has the same domain [1,10]. Let us consider a record belonging to the Leaf 1 of the tree where the logic rule corresponding to Leaf 1 is say, "Uniformity of Cell Size" ≤ 2 AND "Bare Nuclei" ≤ 3. Thus for Leaf 1, the new domain for attribute Uniformity of Cell Size is [1,2], while the new domain for the attribute Bare Nuclei is [1,3]. Recall that when perturbing the influential attributes in our Framework we ensure that the perturbed values remain within the boundaries of the leaf; in this case, the perturbed value of Uniformity of Cell Size remains in the range [1,2], while the new value for Bare Nuclei remains in the range [1,3].

We assumed that the intruder knows the probability distribution of added noise. However, he/she does not know the leaf of the original tree to which the target record x belongs. The best the intruder can do in order to estimate P_{xi}^j is to run the target record through the decision tree built on the perturbed data set and adopt the LINFA and LINNA domains for the leaf in which the target record arrives.

6.3. Entropy (level of security) Calculation of a Perturbed Data Set

From the estimated P_{xi}^j values an intruder can calculate the probability P_{xi} for each and every record of the perturbed data set. We use these P_{xi} probabilities to calculate the entropy of the whole perturbed data set with respect to re-identification of a target record x as follows:

$$\begin{aligned} H_x^{RI} &= -\sum_{i=1}^{n} P_{xi} log_2 P_{xi} \\ &= -\sum_{i=1}^{n} \left(\frac{\prod_{j=1}^{m} P_{xi}^j}{\sum_{i=1}^{n}(\prod_{j=1}^{m} P_{xi}^j)}\right) log_2 \left(\frac{\prod_{j=1}^{m} P_{xi}^j}{\sum_{i=1}^{n}(\prod_{j=1}^{m} P_{xi}^j)}\right) \end{aligned} \tag{5}$$

We next estimate the overall security or, alternatively, disclosure risk of the whole perturbed data set, with respect to all original records. We calculate H_x^{RI}, $x = 1, 2, \ldots \ldots n$, i.e., the entropy of the perturbed data set with respect to each record of the original data set. We then calculate the mean entropy, $\mu = \frac{\sum_{x=1}^{n} H_x^{RI}}{n}$ and the standard deviation, $\sigma = \sqrt{\frac{\sum_{x=1}^{n}(H_x^{RI}-\mu)^2}{n}}$.

The higher entropy H_x^{RI} indicates the higher security and the lower disclosure risk. The higher value of μ generally implies the higher level of overall security of the data set. The overall security can also be estimated as follows. Let T be the percentage of total number of original records that have H_x^{RI} lower than a user defined threshold H_t^{RI}. If $T \leq v$, where v is a user defined threshold then we consider the data set as secure for T, H_t and v. We can also use μ, σ, T, v and H_t^{RI} to have a more detailed report on the security level of a data set.

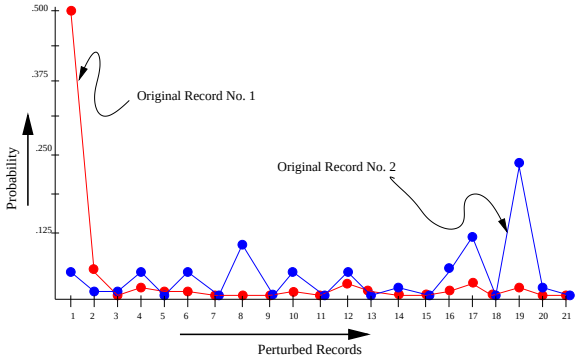

Figure 3: The probability distribution of a perturbed record originating from the target record x.

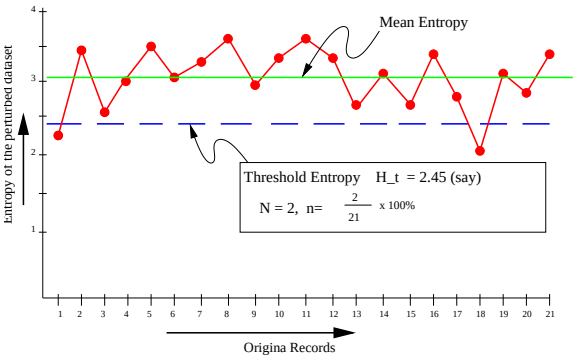

Figure 4: Entropies of a perturbed data set calculated for each original record.

We now illustrate the basic concepts of our security measure with an example and a couple of figures. Suppose we have an original and a perturbed data set, each of them having 21 records. Figure 3 shows a possible probability distribution of a perturbed record originating from the target record

x. According to the figure, the probability that the perturbed record 1 originated from the original record 1 is 0.500, which is the maximum value for original record 1. Similarly, the maximum probability for the original record 2 corresponds to the perturbed record 19. From each of these distributions we calculate the entropy of the perturbed data set (Figure 4). The dashed line shows H_t^{RI} and the nodes/circles represent the entropy value of the perturbed data set calculated against each original record. For a user defined threshold of $H_t^{RI} = 2.45$ there are 2 out of 21 original records (record 1 and record 18) for which the entropies are less than H_t^{RI}.

6.4. Class Value Disclosure Risk Analysis

We next explore the risk of intruder learning the class of the target record. We consider a generalised scenario where an intruder is interested to check if an individual has a class value belonging to a subset of the domain of the class attribute. We present an example scenario as follows. An employer may have an interest to learn if an employee has a serious disease/health problem, which needs long, continuous and expensive treatment. The employer may have a list of diseases/ health problems which he/she considers serious. We also assume that the employer has access to a perturbed data set (having the record that belongs to the particular employee) where the class attribute is "diagnosis".

We assume that in such a scenario an intruder first attempts to identify the target record and then learns the confidential class value. Therefore, in order to define the security level of a data set we first estimate an entropy H_{xk}^c for a target record x having a class value $k \in L = \{l_1, l_2, ...l_s\}$, where $\{l_1, l_2, ...l_s\} \subseteq (D)$, and D is the domain of the class attribute. The probability that a record x has the class value in the set L is as follows.

$$P_x^c(L) = \sum_{i=1}^{n} \left(\frac{\prod_{j=1}^{m} P_{xi}^j}{\sum_{i=1}^{n} (\prod_{j=1}^{m} P_{xi}^j)} \times \sum_{k=l_I}^{l_z} P_i^c(k) \right) \tag{6}$$

where, $P_i^c(k)$ is probability that the value of the class attribute of the *ith* record of the perturbed data set is equal to $k \in L$. Note that the probability $P_i^c(k)$ can be estimated from a decision tree built on the perturbed data set. We simply need to consider the leaf containing the record i and the number of records for each value of the class. For example, let record i belong to the leaf $L5$, where there are n_k records with class value k, $k \in [1, t]$. Then

$P_i^c(k) = \frac{n_k}{\sum_{i=1}^{t} n_i}$. We emphasise again that while this is the best estimate the intruder can make, it is not necessarily the correct value of the probability, as the original and perturbed decision tree may differ to some extent.

The corresponding entropy $H_x^c(L)$, for an intruder learning the class of the target record, can be estimated as follows.

$$H_x^c(L) = -P_x^c(L)log_2P_x^c(L) - P_x^c(D\backslash L)log_2P_x^c(D\backslash L) \quad (7)$$

If $H_x^c(L)$ is greater than a user defined thresholds H_t^c, then we define the perturbed data set as secure for the target record x with respect to L and H_t^c. In order to check the overall security of the perturbed data set with respect to L and H_t^c, we estimate $H_x^c(L)$ for each record belonging to an original data set. We then count b, the number of original records, where $H_x^c(L) < H_t^c$. If $\frac{b}{n}$ (where n is the total number of records in the original data set) does not exceed a user defined threshold v then we consider the data set as secure with respect to L, H_t and v.

6.5. An Illustration of the Security Evaluation

We now illustrate H_x^{RI} and H_x^c calculation of a target record for a perturbed *WBC* data set. We randomly select a record and calculate both H_x^{RI} and H_x^c for the record. In order to evaluate the security level of the perturbed data set we compare them with the corresponding entropies for the case where the intruder has access to the unperturbed data set. In Table 5 a row label corresponds to the number of non-class attributes known to the intruder and the columns correspond to the entropies for the cases where the intruder has access to the original data set and the perturbed data set.

From Table 5 we see that when an intruder has access to original data set he/she can precisely learn the class if he/she knows at least two attribute values for the target record. An intruder can also precisely identify the record and learn all the other attribute values including class whenever he/she knows 5 or more attribute values for the target record.

However, if the intruder has access only to the perturbed data set he/she will never be able to re-identify the record or the class with certainty even if he/she knows all 9 non-class attributes. Moreover, the entropy for re-identification remains high regardless of the number of attribute values known to the intruder. For example, even if an intruder knows all 9 attribute values of the target record the entropy for re-identification of record is 6.643 which roughly corresponds to the case where there are 100 equally likely records to choose from.

Number of attributes known	Original		Perturbed	
	$H^{c}_{1321264}$	$H^{RI}_{1321264}$	$H^{c}_{1321264}$	$H^{RI}_{1321264}$
9	0	0	0.311	6.643
8	0	0	0.34	6.78
7	0	0	0.434	7.328
6	0	0	0.56	7.689
5	0	0	0.747	7.794
4	0	2.322	0.835	7.988
3	0	2.585	0.86	8.199
2	0	3	0.955	8.404
1	0.503	5.17	0.970	8.676
0	0.931	9.229	0.931	9.229

Table 5: Privacy preserving techniques - comparative study

What we consider the most valuable property of our noise addition technique is the fact that we can always adjust the level of noise we are adding to the attributes so as to achieve the desired level of the security. The only attributes we can not perturb beyond the limits given by the leaves are the leaf influential attributes. However, we argue if leaf influential attributes are known to the intruder then he/she can run the record through the decision tree to learn the class rather than attempt to re-identify the record. He/she will not however be able to learn other attributes in addition to the class attribute as the re-identification entropy remains high regardless of intruders knowledge of influential attributes. Our method works best for dense data sets such as *WBC*.

7. Conclusion

In this paper we have introduced a *Framework*, that incorporates several novel techniques to perturb all attributes of a data set. Our experimental results indicate that the *Framework* is very effective in preserving original patterns in a perturbed data set. The trees obtained from data sets perturbed by the *Framework* are very similar to the original tree. For Adult data set, four out of five perturbed trees have 100% Type A or Type B rules. The fifth tree has 70% Type A or Type B rules and 0% Type D rules. Moreover, for

WBC data set all five trees have $\geq 90\%$ Type A or Type B rules. Framework performs better than all other techniques in terms of preservation of similarity of trees, for both data sets (see Table 1 and Table 3).

Prediction accuracies of trees obtained from data sets perturbed by *Framework* are better than prediction accuracies of trees obtained from data sets perturbed by *Random Framework* and *Random Extended Framework*, for both data sets. Moreover, it is better than *Extended Framework* for Adult data set (see Table 2 and Table 4). However, *Framework* does not maintain the original correlations among attributes very well.

Therefore, in order to also preserve the original correlations in a perturbed data set in this study we have presented *Extended Framework* that incorporates *GADP* [3]. Our experimental results suggest that the *Extended Framework* preserves original correlations significantly better (in terms of $AvgDifValue$ and $AvgDifMat$ values) than how it is preserved by the Framework and Random Framework, for both data sets.

However, we believe that the *Extended Framework* can preserve original correlations even better if *C-GADP* or *EGADP* is used instead of *GADP*, since both (Adult and WBC) data sets do not have multivariate normal distribution. Moreover, *GADP* has been applied on small sets of records belonging to the leaves separately. Note that *GADP* preserves the correlations very well when it is applied on a large data set having multivariate normal distribution.

Extended Framework also preserves a high data quality in terms of prediction accuracy of the classifiers obtained from perturbed data sets. It performs better than all techniques other than *Framework* in terms of prediction accuracy, for Adult data set. However, for WBC data set trees obtained from Extended Framework perturbed data sets have around 2% better accuracy than even the original trees (see Table 2 and Table 4). It performs better than all techniques including Framework, for WBC data set. Additionally, it preserves the original patterns significantly better than the patterns preserved by *Random Framework* and *Random Extended Framework*.

It is evident from our experiments that *GADP* preserves the original correlations very well, but it does not perform well in terms of decision tree similarity (see Table 3). The sizes (number of leaves) of the perturbed trees are 31, 26, 10, 34 and 37, respectively while the size of the original tree is only 14. However, the application of *GADP* on the records belonging to each leaf separately according to our *Extended Framework* preserves original patterns significantly better than the use of *GADP* on the whole data set at

a time. This also justifies the reason why we need separate noise addition techniques (such as the *Framework* and the *Extended Framework*) catered for data mining when we already have techniques (such as *GADP*) tailored for statistical data analysis. Our future work plan includes experiments on our *Extended Framework* using *CGADP* and *EGADP*, and designing improved perturbation techniques.

Acknowledgement:

This work was supported by the ARC Grant No DG-DP0452182, and Seed Grant, Faculty of Business, Charles Sturt University, Australia.

References

[1] W. C. G. P. Ltd, Community attitude towards privacy 2007, a survey prepared for the office of the federal privacy commissioner, australia, survey prepared by roy morgan research, available from www.privacy.gov.au/materials/types/download/8820/6616, 2007. Visited on 19.10.09.

[2] Y. Zhu, L. Liu, Optimal randomization for privacy preserving data mining., in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2004, Seattle, Washington, USA, pp. 761–766.

[3] K. Muralidhar, R. Parsa, R. Sarathy, A general additive data perturbation method for database security, Management Science 45 (1999) 1399–1415.

[4] R. Sarathy, K. Muralidhar, R. Parsa, Perturbing non-normal confidential attributes: The copula approach, Management Science 48 (2002) 1613–1627.

[5] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: Proc. of the ACM SIGMOD Conference on Management of Data, ACM Press, 2000, pp. 439–450.

[6] V. Estivill-Castro, L. Brankovic, Data swapping: Balancing privacy against precision in mining for logic rules., in: Proc. of Data Warehousing and Knowledge Discovery (DaWaK 1999).

[7] W. Du, Z. Zhan, Using randomized response techniques for privacy-preserving data mining., in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2003, Washington, DC, USA, pp. 505–510.

[8] S. R. M. Oliveira, O. R. Zaïane, Algorithms for balancing privacy and knowledge discovery in association rule mining, in: Proc. of the 7th International Database Engineering and Applications Symposium (IDEAS 2003), Hong Kong, China, p. 5463.

[9] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, E. Dasseni, Association rule hiding., IEEE Trans. Knowl. Data Eng. 16 (2004) 434–447.

[10] S. Rizvi, J. R. Haritsa, Maintaining data privacy in association rule mining., in: Proceedings of the 28th VLDB Conference 2002, Hong Kong, China, pp. 682–693.

[11] S. R. M. Oliveira, O. R. Zaïane, Protecting sensitive knowledge by data sanitization., in: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), Melbourne, Florida, USA, pp. 613–616.

[12] S. R. M. Oliveira, O. R. Zaïane, Foundations for an access control model for privacy preservation in multi-relational association rule mining, in: Proceedings of IEEE ICDM Workshop on Privacy, Security and Data Mining 2002, Maebashi City, Japan, pp. 19–26.

[13] S. R. M. Oliveira, O. R. Zaïane, Privacy preserving frequent itemset mining, in: Proceedings of IEEE ICDM Workshop on Privacy, Security and Data Mining 2002, Maebashi City, Japan, p. 4354.

[14] N. Zhang, S. Wang, W. Zhao, A new scheme on privacy preserving association rule mining., in: Knowledge Discovery in Databases: PKDD 2004, 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, pp. 484–495.

[15] A. Schuster, R. Wolff, B. Gilburd, Privacy-preserving association rule mining in large-scale distributed systems., in: Proceedings of 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, Illinois, USA, pp. 411–418.

[16] Y. Saygin, V. S. Verykios, A. K. Elmagarmid, Privacy preserving association rule mining., in: RIDE 2002, pp. 151–158.

[17] A. V. Evfimievski, R. Srikant, R. Agrawal, J. Gehrke, Privacy preserving mining of association rules., in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2002, pp. 217–228.

[18] L. Brankovic, V. Estivill-Castro, Privacy issues in knowledge discovery and data mining., in: Proc. of Australian Institute of Computer Ethics Conference (AICEC 1999), Melbourne, Victoria, Australia.

[19] R. Sarathy, K. Muralidhar, R. Parsa, Perturbing nonnormal confidential attributes: The copula approach, Management Science 48 (2002) 1613–1627.

[20] K. Muralidhar, R. Sarathy, An enhanced data perturbation approach for small data sets, Decision Sciences 36 (2005) 513–529.

[21] M. Z. Islam, L. Brankovic, Detective: A decision tree based categorical value clustering and perturbation technique in privacy preserving data mining, in: Proceedings of the 3rd International IEEE Conference on Industrial Informatics (INDIN 2005), Perth, Australia.

[22] J. R. Quinlan, Improved use of continuous attributes in c4.5, Journal of Artificial Intelligence Research 4 (1996) 77–90.

[23] M. Z. Islam, Explore: A novel decision tree classification algorithm, in: Proceedings of the 27th International Information Systems Conference, British National Conference on Databases (BNCOD 2010), Springer LNCS 6121 (in press), Dundee, Scotland.

[24] M. Z. Islam, L. Brankovic, Noise addition for protecting privacy in data mining, in: Proceedings of of the 6th Engineering Mathematics and Applications Conference (EMAC 2003), volume 2, Sydney, Australia, pp. 85–90.

[25] M. Z. Islam, Privacy Preservation in Data Mining through Noise Addition, Ph.D. thesis, School of Electrical Engineering and Computer Science, The University of Newcastle, Australia, 2008.

[26] M. Z. Islam, L. Brankovic, A framework for privacy preserving classification in data mining., in: Proceedings of Workshop on Data Mining and Web Intelligence (DMWI 2004), pp. 163–168.

[27] V. Ganti, J. Gehrke, R. Ramakrishnan, Cactus - clustering catagorical data using summaries, in: Proceedings of ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining 1999, San Diego, CA, USA.

[28] U. U. of California, Uci machine learning repository of ucirvine university of california, in: available from http://www.ics.uci.edu/∼mlearn/MLRepository.html. Visited on 12.10.06.

[29] P. Fule, J. F. Roddick, Detecting privacy and ethical sensitivity in data mining results., in: V. Estivill-Castro (Ed.), Proceedings of 27th Australian Computer Science Conference (ACSC 2004), volume 26 of *Conference in Research and Practice in Information Technology*, pp. 163–168.

[30] J. Roddick, P. Fule, A System for Detecting Privacy and Ethical Sensitivity In Data Mining Results, Technical Report SIE-03-001, School of Informatics and Engineering, Flinders University, Adelaide, Australia, 2003.