DETECTIVE: A Decision Tree Based Categorical Value Clustering and Perturbation Technique for Preserving Privacy in Data Mining

Md. Zahidul Islam and Ljiljana Brankovic

Md. Zahidul Islam and Ljiljana Brankovic, Electrical Engineering & Computer Science, The University of Newcastle, Callaghan, NSW 2308, Australia, e-mail : (zahid, lbrankov) @cs.newcastle.edu.au

Abstract— **Data Mining is a powerful tool for information discovery from huge datasets. Various sectors, including commercial, government, financial, medical, and scientific, are applying Data Mining techniques on their datasets that typically contain sensitive individual information. During this process the datasets get exposed to several parties, which can potentially lead to disclosure of sensitive information and thus to breaches of privacy.**

Several Data Mining privacy preserving techniques have been recently proposed. In this paper we focus on data perturbation techniques, i.e., those that add noise to the data in order to prevent exact disclosure of confidential values. Some of these techniques were designed for datasets having only numerical non-class attributes and a categorical class attribute. However, natural datasets are more likely to have both numerical and categorical non-class attributes, and occasionally they contain only categorical attributes. Noise addition techniques developed for numerical attributes are not suitable for such datasets, due to the absence of natural ordering among categorical values. In this paper we propose a new method for adding noise to categorical values, which makes use of the clusters that exist among these values. We first discuss several existing categorical clustering methods and point out the limitations they exhibit in our context. Then we present a novel approach towards clustering of categorical values and use it to perturb data while maintaining the patterns in the dataset. Our clustering approach partitions the values of a given categorical attribute rather than the records of the datasets; additionally, our approach operates on the horizontally partitioned dataset and it is possible for two values to belong to the same cluster in one horizontal partition of the dataset, and to two distinct clusters in another partition. Finally, we provide some experimental results in order to evaluate our perturbation technique and to compare our clustering approach with an existing method, the so-called CACTUS.

Index Terms— **clustering, data perturbation, privacy, privacy preserving data mining.**

I. INTRODUCTION

Due to the development of information processing technology and storage capacity huge amount of data is being collected and processed in almost every sector of life. Supermarket chains and departmental stores typically capture each and every sale transaction of their customers. For example, Wal-Mart Stores Inc. captures sale transactions from more than 2,900 stores in 6 different countries and continuously transmits these data to its massive data warehouse, which is the biggest in retail industry, if not the biggest in the world [2, 6, 15]. These data are subjected to Data Mining techniques in order to extract useful information, which is in turn used for various purposes such as marketing of products and services, identifying the best target group/s, and improving business strategies. Wal-Mart allows more than 3,500 suppliers to access these data and performing data analysis. According to Teradata, Wal-Mart has plans to expand its huge warehouse to even huger, allegedly to a capacity of 500 terabytes [2].

Sometimes collected data are used for a secondary purpose, that is, a purpose for which they were not initially collected. During the whole process of Data Mining (from collection of data to discovery of knowledge) these data, which typically contain sensitive individual information, often get exposed to several parties. Disclosure and intentional misuse of such sensitive information can cause serious breach of individual privacy. For example, detailed credit card record of an individual can expose the private life style with sufficient accuracy.

In 2004, the Office of the Federal Privacy Commissioner, Australia, engaged Roy Morgan Research to investigate community attitude towards privacy [5]. According to the survey, 81% of the respondents believe that "customer details held by commercial organisations are often transferred or sold in mailing lists to other businesses". Additionally, 94% of the respondents consider acquisition of their personal information by a business they do not know as a privacy invasion. Secondary use of personal information by the collecting organisation is considered as a privacy invasion by 93% of the respondents. Respondents were most reluctant to disclose details about finances (41%) and income (10%).

Public awareness of privacy and lack of public trust in organisations may introduce additional complexity to data collection. As a result, organisations may not be able to fully enjoy the benefits of various Data Mining techniques. This may affect their ability for strategic planning for marketing, production, etc. Therefore, suitable privacy preserving techniques have become vital in Data Mining. Researchers and organisations have proposed many data perturbation techniques [1, 7, 11, 12, 13]. However, most of the existing techniques consider a dataset with all numerical non-class attributes. Natural datasets are more likely to have both categorical and numerical non-class attributes, and occasionally they contain only categorical non-class attributes. Due to the absence of natural ordering in categorical values data perturbation techniques for numerical

This work was supported by the ARC Grant No DG-DP0452182.

data perturbation techniques for numerical values can not be applied to categorical values.

We believe that the natural way of overcoming the absence of ordering in categorical values is to cluster them. Values within the same cluster can be treated as more similar to each other. We also need a similarity measure so as to establish if a value pair is more similar than another value pair within the same cluster. Existing algorithms for categorical clustering group records [3,4,8,9,10]. However, in this study we are interested in clustering categorical values belonging to the same attribute. One possible approach to achieve this is to use the existing clustering methods and consider values (belonging to an attribute) to be similar if they appear in the same cluster of records. Another approach is to directly cluster the attribute values without relying on clusters of records. In this paper we use the second approach.

The organisation of the paper is as follows. In Section 2 we discuss some existing clustering techniques and their limitations in the context of noise addition. In Section 3 we introduce a novel approach towards clustering categorical values belonging to an attribute. We present a perturbation technique which incorporates the clustering concept. In Section 4 we present some experimental results to demonstrate advantages of our technique. Finally, we present our concluding remarks in Section 5.

II. Existing Clustering Techniques

In this section we discuss some existing techniques for clustering datasets with categorical attributes and we expose some of their limitations which make them unsuitable for our purpose.

Ganti, Gehrke and Ramakrishnan [8] proposed CACTUS, which clusters Cartesian product of attribute domains. It indirectly defines a cluster to be a collection of records such that every pair of values (belonging to two different attributes) within a cluster is strongly connected. Two values (belonging to two different attributes) are considered to be strongly connected if they appear together in δ times more than the expected number of records, where δ is a user defined constant. The expected number is calculated based on the assumption that the attributes are independent, and that all their values are equally likely (the so called Attribute Independence Assumption). CACTUS regards two values from the same attribute as similar with respect to another attribute, if both values are strongly connected with a value of the other attribute.

CACTUS consists of three phases: summarisation, clustering and validation. In summarisation phase it first builds inter-attribute summaries comprising all pairs of attribute values which are strongly connected. It then builds intra-attribute summaries which contain all similar pairs of values belonging to each attribute. In the clustering phase CACTUS first produces all cluster projections on each attribute. It then creates candidate clusters on all the attributes from the cluster projections on individual attributes. Finally, in the validation phase CACTUS computes the set of actual clusters from the set of candidate clusters. If the support of a candidate cluster is greater than a threshold then that candidate cluster is considered to be an actual cluster. Support of a candidate cluster is measured by the number of records that belong to the candidate cluster. According to CACTUS, an attribute value may belong to several clusters. However, a record can only belong to one cluster.

We observe a few characteristics of CACTUS which are not appropriate in the context of this study. CACTUS considers the whole dataset in order to measure if two values (belonging to two different attributes) are strongly connected. However, we argue that two values may not be found strongly connected within the whole dataset, but they can still be strongly connected in a particular segment of the dataset. Additionally, the Attribute Independence Assumption used in CACTUS may not be realistic [4]. Furthermore, the definition of a cluster used in CACTUS is overly restrictive and hence CACTUS may end up with huge number of clusters [4]. We also observe that the presence of an attribute which is not correlated with other attributes may drastically increase the number of clusters, in the case where the values in the all or at least some attributes are approximately equally likely.

Chuang and Chen [4] proposed a correlation based agglomerative hierarchical clustering algorithm called CORE, which explores the correlation between attribute values in order to extract clusters of records. CORE considers a record as a set of attribute values that comprise the record. Since CORE clusters records, each record can only belong to one cluster. However, any value of an attribute may appear in several clusters.

CORE computes correlations for every pair of attribute values (belonging to the same or different attributes) with a *Jaccard coefficient measure* as a ratio of the number of records having both of the values, to the number of distinct records having any of the values. CORE then computes the so called Force Correlation for every pair of values by deducting a user defined constant from the correlation for corresponding pair of values. If the Force Correlation is positive then the corresponding attribute values are considered to be attractive, otherwise repulsive.

CORE first considers each record as a cluster. It then computes similarity for each pair of clusters based on their Force Correlation. Two clusters with the largest similarity are merged together. CORE then goes into the next hierarchical merging iteration. The merging iteration continues in a loop until a termination condition is satisfied.

We observe a feature of CORE which is not appropriate in the context of this study. Recall that CORE calculates correlation between two values (belonging to same or different attributes) as a ratio of the number of records with both of the values, to the number of records with either of the values. Our concern is that in such measure the correlation between two values will be small if one appears in a very large number of records and the other appears in a small number of records. However, if the second value always appears with the first one then the second value is actually highly correlated with the former value. CORE over-

looks such strong correlation.

Guha, Rastogi and Shim [9] proposed an agglomerative hierarchical clustering algorithm called ROCK in the context of Market Basket dataset. ROCK measures similarity of two records by the ratio of the number of the attribute values appearing in both records to the number of distinct attribute values appearing in any of the records. It considers two records as neighbours if their similarity exceeds a given threshold. It next assigns links between every two neighbour records where number of links between them is the same as the number of their common neighbours. ROCK aims to maximize the number of links within same cluster and minimize the number of links across different clusters. ROCK clusters records where each record belongs to only one cluster. However, an attribute value may appear in several clusters.

ROCK requires the desired number *k* of clusters as an input. It then draws a random sample (of size bigger than *k*) from the dataset. Initially it considers each record of the sample as a cluster. It next computes links for each pair of records of the sample. Based on these links, ROCK subsequently computes the so-called "goodness measure" for each pair of clusters. The pair with the maximum "goodness measure" is then merged together. After that the "goodness measures" of the new cluster with all other clusters are calculated. ROCK then goes into the next hierarchical merging iteration. The merging continues until the records of the sample are clustered in *k* different clusters. Finally ROCK assigns each of the remaining records to the most appropriate cluster among these *k* clusters. ROCK is considered to be an outstanding algorithm for clustering categorical values [4]. However, the user defined threshold for neighbours' similarity is difficult to determine without prior knowledge of the dataset [4]. We note that with a low threshold, a large number of links between two records may be established without indicating sufficient similarity between them.

In 2002 Barbara, Couto and Li [3] proposed COOLCAT, the entropy based heuristic algorithm for clustering records. COOLCAT is based on the fact that entropy of a cluster containing similar records will be lower than the entropy of a cluster containing dissimilar records. They aimed to minimize the expected entropy of the whole arrangement. COOLCAT clusters records of a dataset into *k* non overlapping clusters, where *k* is a user defined number.

COOLCAT consists of two phases: initialisation phase and incremental phase. In the initialisation phase it first draws a random sample from the dataset. It then finds the two records belonging to the sample which maximise the entropy. These two records are placed in two separated clusters. It then picks the third record which maximises the minimum pairwise entropy with the two existing clusters. This third record is then considered as the third cluster. COOLCAT continues the iteration until it creates *k* most dissimilar clusters out of the records of the sample. In the incremental phase each of the remaining records of the dataset is assigned to one of the *k* clusters such that the expected entropy of the whole arrangement is minimised at every step.

III. A Novel Noise Addition Technique

A few privacy preserving classification techniques were proposed in [7,11,12,13]. In those techniques a small amount of noise was added to a dataset with numerical non-class attributes and a categorical class attribute. However, in case of categorical attribute values, due to the absence of any natural ordering it is not clear how to add a small amount of noise. In our approach we first cluster categorical attribute values and then use the clusters to add noise. In the following paragraphs we introduce our categorical attribute perturbation technique in details.

We consider a dataset with both numerical and categorical non-class attributes along with a categorical class attribute. We primarily consider the categorical class attribute as sensitive and confidential. We then assume that the dataset is required to be released for various purposes such as research, marketing, health care, and finance. Ideally, the whole dataset should be released without any access restriction so that users can apply various data mining techniques. We assume that the extraction of general patterns and properties does not constitute a breach of privacy. However, disclosure of confidential individual values with sufficient certainty is indeed a breach of privacy. Deletion of identifiers such as name, customer ID, or credit card number is not sufficient to prevent such disclosure. A user with sufficient supplementary knowledge about an entity may still be able to identify an individual, and then learn the confidential class attribute value. Perturbation of all attributes instead of just one confidential attribute can help attain the same level of privacy with less amount of noise on each attribute due to the distribution of noise over all attributes [11,13]. In this study we further explore these ideas.

Most existing techniques measure similarity of two values (belonging to a categorical attribute) based on their co-occurrence with values belonging to another categorical attribute. In that respect, the underlying idea of our clustering technique is similar to the basic concept of existing techniques.

Our clustering technique differs from existing techniques on few aspects. Firstly, our technique takes values belonging to a numerical attribute into account in measuring similarity of two categorical values. If both categorical values co-occur with values of a numerical attribute that fall within the same range, then the categorical values are considered to be similar. Thus our technique is directly applicable to a dataset having both numerical and categorical attributes, unlike most existing techniques which are suitable for categorical attributes only.

Secondly, unlike most other clustering techniques, our method divides the dataset in appropriate horizontal partitions, and clusters values of a categorical attribute within each horizontal partition of the dataset instead of within the whole dataset. We note that two attribute values may be considered very similar within a particular segment of the dataset while they can not be considered similar in the data-

set as a whole. For example, nursing and secretarial jobs can be considered similar among people older than 50 years, because of the fact that both of these jobs were typically performed by females in the past. However, since these jobs are nowadays performed by both males and females, there is no similarity between them among younger part of the population. Another advantage of our technique is its focus on few relevant attributes, instead of all the attributes. By "relevant attributes" we mean those attributes which have high correlation with the concerned attribute, and which explain the attribute the best.

Before we introduce our technique in more detail, we first describe a few ideas and provide a few definitions. Our technique makes use of decision trees for clustering and perturbing categorical attribute. In order to perturb a categorical attribute, say attribute A, we first build a decision tree that considers attribute A as the class attribute. In a decision tree, each node tests a value of a particular attribute. If the attribute is categorical then the corresponding node typically has as many children as there are values in the domain of the attribute. This corresponds to dividing the dataset, or a segment of it, into as many partitions as there are values in the domain of the attribute. However, if the attribute is numerical then the dataset is divided into two partitions. Thus, if the node tests a categorical attribute, then the records in each partition contain identical values of the attribute, otherwise they contain values that fall within the same range. Each leaf of the tree represents a horizontal segment of the dataset.

In what follows we refer to the path from the root to a leaf as the leaf-path for that particular leaf. Attributes tested on the nodes of a leaf-path are called Leaf Influential Attributes (LINFAs) of the leaf [11]. Remaining attributes of the dataset are called Leaf Innocent Attributes (LINNAs) of the leaf. Additionally, attributes of the dataset which are not tested in any of the nodes of the decision tree are called Total Innocent Attributes (TIAs). Thus, records belonging to a leaf have the same value in each categorical LINFA and the values falling in the same range for each numerical LINFA. Among all the attributes of the dataset LINFAs are the relevant attributes as they best describe the class attribute value of the leaf. The essence of our method is to horizontally partition (cluster) the dataset into leaves and collections of leaves and consider two categorical values of the same attribute similar if they belong to the same partition. We note that in general, for each categorical attribute we get different dataset partitions (clusters). Our method has two main characteristics: Firstly, for measuring similarity between categorical values we only consider relevant attributes. Secondly, similarity is defined within a certain horizontal partition only.

Two leaves are called siblings if their leaf-paths differ only in the last node. Any two records belonging to two sibling leaves have all corresponding categorical values the same and numerical values falling in the same range for all the LINFAs except for the LINFA tested on the last node. Hence class attribute values of the records belonging to two sibling leaves are considered to be similar.

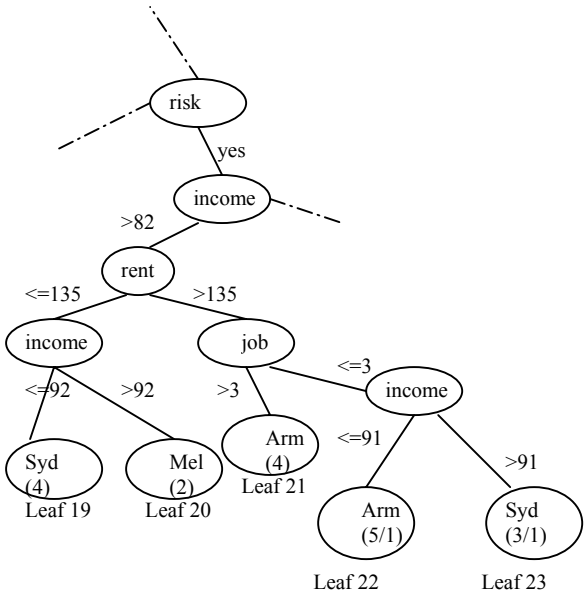

Fig.3.1 A Section of the decision tree built on *CRD* dataset. The tree considers attribute *City* as class attribute.

Furthermore, a leaf is called homogeneous if all records belonging to the leaf contain same class value. Otherwise the leaf is called heterogeneous. The predominant class value of the records in a heterogeneous leaf is refered to as the majority class of the leaf. Any other class value that appears at least in one record of a heterogeneous leaf is called a minority class. We argue that class values belonging to the records of a heterogeneous leaf are similar due to fact that they share the same values in LINFAs. Indeed, these values are so similar that they can not easily be further separated.

We now illustrate these ideas and definitions on a few examples [Fig. 3.1]. We introduce a synthetically created dataset called Credit Risk Dataset (*CRD*). The *CRD* has four numerical and one categorical non-class attributes, and a categorical class attribute. The numerical attributes are *Income*, *House Rent*, *No. of Dependents*, and *Job Grade*. Their domains are respectively [30, 100], [100, 600], [0, 7], and [1, 4]. The only categorical non-class attribute is *City*. The domain of *City* is {Sydney, Melbourne, Newcastle, Armidale}. The categorical class attribute is *Credit Risk*, the domain of which is {yes, no}.

In order to cluster values of attribute *City* we create a decision tree which considers *City* as the class attribute. In Fig 3.1 we present a section of the decision tree. There are six internal nodes and five leaves in this section of the decision tree. Leaf number 22 and leaf number 23 are examples of heterogeneous leaves, and the other leaves are homogeneous. For example, there are five records in leaf 22; in attribute *City* , four of these records have value "Armidale", and the remaining record has a different value. Therefore majority class of the leaf is "Armidale". Leaf 19 and leaf 20 are siblings. They have exactly the same leaf-path except for leaves themselves. Their LINFAs include *Credit Risk*, *Income*, and *House Rent*. Similarly, leaf 22 and leaf 23 are

siblings.

Within the horizontal segment defined by leaves 19 and 20, class values "Sydney" and "Melbourne" are considered to be similar. Likewise, "Sydney" and "Armidale" are similar within leaves 22 and 23. Moreover, the majority and minority classes of a heterogeneous leaf are considered "more similar" than the classes of the sibling leaves.

We next explain our technique, DETECTIVE (DEcision TreE based CaTegorIcal ValuE clustering and perturbation technique) in detail. In order to perturb an attribute A, DETECTIVE first considers the attribute A as a class and creates a corresponding decision tree. DETECTIVE next considers each leaf of the tree in turn. For a leaf L with y siblings, DETECTIVE calculates values l_i, $1 \leq i \leq y$; additionally, for each heterogeneous leaf with x minority classes, DETECTIVE calculates value q, as follows. Let N be the total number of records in the leaf L. Out of these N records, let m belong to the majority class of the leaf, and let n_i , $1 \leq i \leq x$ belong to the i^{th} minority class. DETECTIVE now calculates q and l_i, $1 \leq i \leq$ y as follows: $q=m/N$ and $l_i = n_i/N$. DETECTIVE next takes as an input a user defined value p that determines the value of noise added. For each record of the leaf L, DETECTIVE changes the class to the majority class of one of the siblings with the probability p/y. Furthermore, if L is heterogeneous leaf, then every record is assigned the majority class with the probability q $(1-p)$, and the i^{th} minority class with the probability l_i $(1-p)$.

If a leaf has no siblings, then p is considered to be 0 and the class is "shuffled" within the leaf with probability 1. Note that class values in homogeneous leaves without siblings are not perturbed at all. If higher value of security is required, then the class values can be changed not only to classes of sibling leaves but also to classes of "cousin " leaves that are further apart in the decision tree. Alternatively, the class can be changed to any other value randomly with a user defined probability v. The drawback of both of these approaches is of course higher distortion of the patterns in the dataset.

As an illustration we apply DETECTIVE on the *CRD* dataset in order to perturb the values of attribute *City*. DETECTIVE first creates a decision tree that considers attribute *City* as class attribute. It then scans the records one by one. For each record, DETECTIVE identifies the leaf which contains the record. For example, consider a record that belongs to the leaf 22. Leaf 22 is a heterogeneous leaf, and the majority class is "Armidale" while the minority class is "Sydney". Additionally, leaf 22 has a sibling, which is the leaf 23. Majority class of leaf 23 is "Sydney". DETECTIVE changes the class value (that is, the value for attribute *City*) of the record to the majority class of the sibling leaf which is "Sydney" with a user defined probability p. Furthermore, since leaf 22 is a heterogeneous leaf it shuffles the class values of the records belonging to the leaf with a probability $(1 - p)$.

Let us now consider another example where a record belongs to leaf 19, which is a homogeneous leaf having a sibling. In such case DETECTIVE changes the class value of the record ("Sydney") to the class value of the sibling leaf ("Melbourne") with a user defined probability p. However, with a probability $(1-p)$ it leaves the class value of the record unchanged.

DETECTIVE perturbs a non-class categorical attribute of a dataset. Hence, in order to perturb all non-class categorical attributes we apply it on the original dataset once for every non-class categorical attribute. Each time it produces a dataset with one perturbed attribute in it. Finally, we produce a dataset (combining all perturbed datasets) where each non-class categorical attribute is perturbed and all other attributes are unperturbed.

Application of DETECTIVE along with Leaf Innocent Attribute Perturbation Technique (LINNAPT), Leaf Influential Attribute Perturbation Technique (LINFAPT), and Class Attribute Perturbation Technique [7,11,12,13] results in perturbation of all the attributes of a dataset.

IV. Experimental Results

We perform the experiments in two phases. In the first phase, we apply both DETECTIVE and CACTUS on *CSD* (a synthetic dataset) in order to investigate the quality of DETECTIVE in similarity discovery. Furthermore, in the second phase of the experiments, we apply DETECTIVE on two synthetic datasets *CRD* and *CSD* in order to perturb them. For each of these datasets, we then compare the data quality of the perturbed datasets with the data quality of the original dataset for estimating the effectiveness of DETECTIVE in preserving the patterns. By "data quality" we mean the preservation of patterns of the original dataset. We measure data quality of a perturbed dataset by comparing similarities of the decision trees built on the perturbed dataset and on the original dataset. Again we compare similarities of decision trees based on their logic rules, number of misclassified cases etc. [12]. Our experimental results show that the quality of a perturbed dataset remains satisfactory. We use C5 decision tree builder to create all decision trees in this study.

In the first phase of the experiments we use 399 cases of a synthetically created dataset called *Customer Status Dataset* (*CSD*). *CSD* has five categorical non-class attributes and a categorical class attribute. Five categorical non-class attributes are *Country of Origin*, *Car Make*, *Profession*, *Parents' Country*, and *Favourite City*. The categorical class attribute is *Status*. Domains of the attributes are as follows: *Country of Origin* ={USA, England, Australia}, *Car Make* ={Toyota, Ford, Holden, Nissan}, *Profession* ={Scientist, Engineering, Academic}, *Status* ={Good, Bad}. Domain size of each of the attributes *Parents' Country* and *Favourite City* is thirty. These two attributes randomly draw values from their corresponding domains and do not have any correlation with other attributes.

We apply DETECTIVE on the *CSD* in order to cluster values belonging to the attribute *Car Make*. DETECTIVE creates a decision tree that considers *Car Make* as class attribute [Fig 4.1]. In the decision tree there are two heterogeneous leaves, leaf 1 and leaf 3. Leaf 1 contains 247 records out of which 132 records have "Ford", 62 records

have "Toyota", 48 records have "Nissan", and 5 records have "Holden" for attribute *Car Make*. DETECTIVE discovers that within the segment represented by the leaf 1, "Ford – Toyota" are the most similar. Moreover, "Ford – Nissan" and "Toyota – Nissan" are very similar too. Similarity of each value pair can be measured by multiplying their corresponding number of occurrences. For example, similarity between "Ford – Toyota", "Ford – Nissan", and "Toyota – Nissan" can be measured as 8184, 6336, and 2976 respectively. Similarly, within another heterogeneous leaf, leaf 3 DETECTIVE discovers "Toyota – Holden" to be the most similar. Moreover, there are three sibling leaves – leaf 2, leaf 3 and leaf 4. Within the union of the segments represented by these three leaves it also discovers the similarity of "Toyota – Nissan".

We then apply CACTUS on the *CSD*. We exclude two totally uncorrelated attributes *Parents' Country*, and *Favourite City* from the *CSD* dataset prior to the application of CACTUS, due to our hesitation regarding its performance in presence of such attributes. CACTUS produces a cluster shown in the Table 4.1. From this cluster we learn that "Toyota" and "Ford" are similar within the segment of the dataset represented by the cluster. CACTUS does not discover any other similarity among the values belonging to the attribute *Car Make*. However, we note that DETECTIVE discovers many other similarities in addition to the one CACTUS discovers. Furthermore, it can also provide a similarity measure that evaluates which value pair is more similar than other value pairs.

In the second phase of experiments we use two synthetic datasets *CRD*, and *CSD*. We first use 399 records of the *CRD* dataset. From the original *CRD* dataset, we create a decision tree T_o that considers the natural class attribute *Credit Risk* as the class attribute. We then apply DETECTIVE on the dataset and perturb values for the categorical non-class attribute *City*. After perturbation we create another decision tree T_p (from the perturbed dataset) that also considers the natural class attribute *Credit Risk* as the class attribute. We then compare the similarity of these two decision trees.

We run this experiment ten times. Each time we end up with a decision tree T_p which is extremely similar to the decision tree T_o. Typically, the differences of the T_p with the T_o include slightly different number of records belonging to some leaves, slightly different constant value used for numerical attributes in some nodes, and a different order for testing some of attributes in deep portions of the tree. Nevertheless, for all of the perturbed trees - almost all of the logic rules are exactly the same as the corresponding logic rules of the original tree. Moreover, the number of misclassified cases for any of the perturbed trees is not high. As decision tree algorithms are instable to noise [14], these results suggest the high preservation of data quality in the perturbed datasets.

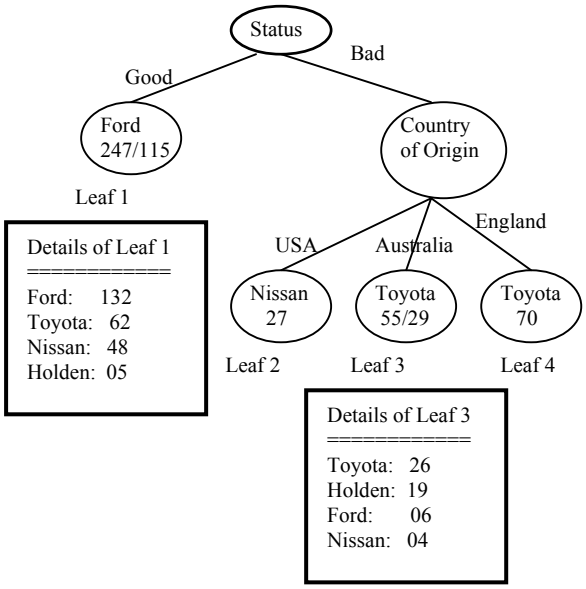

Fig. 4.1 Details of a decision tree built from the unperturbed *CSD*. The tree considers attribute *Car Make* as class attribute. This tree is used for clustering values of attribute *Car Make*.

Country of Origin	Car Make	Profession	Parents' Country	Favourite City	Status
USA	Toyota	Scientist	-	-	Good
Eng.	Ford	Scientist	-	-	Good
USA	Ford	Scientist	-	-	Good
Eng.	Toyota	Scientist	-	-	Good

Table 4.1 The cluster produced by CACTUS from the *CSD*.

We then use 399 cases of the *CSD*. From the original *CSD* we first produce two decision trees, T_o*(status)* and T_o*(car_make)*, that consider attribute *Status* and attribute *Car Make* as class attribute respectively. We then apply DETECTIVE on the *CSD* and perturb only the categorical non-class attribute *Car Make*. We create two decision trees, T_p*(status)* and T_p*(car_make)*, from the perturbed dataset. We compare the similarity of a decision tree built on the perturbed dataset with the corresponding decision tree built on the original dataset.

We also run this experiment ten times. Each time we obtain a decision tree T_p*(car_make)* which is extremely similar to the corresponding decision tree T_o*(car_make)*. All of the logic rules for each T_p*(car_make)* are exactly same as the corresponding logic rules of the T_o*(car_make)*. Moreover, in each of the ten experiments we got a decision tree T_p*(status)* which is very similar to the decision tree T_o*(status)*. Logic rules covering almost all records are similar. Occasionally some logic rules, covering less number of records, differ slightly. Some portions of the perturbed trees are pruned form of the corresponding portion of the original tree. This suggests loss of some minor patterns in the perturbed datasets. However, all major patterns are satisfactorily preserved in all of the perturbed trees.

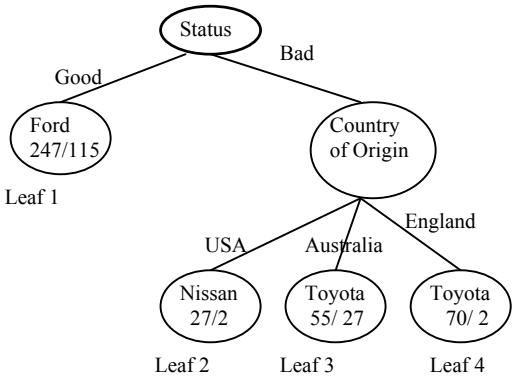

Fig. 4.2 A decision tree built on the total perturbed *CSD* dataset. The tree considers attribute *Car Make* as class attribute.

Finally, we apply DETECTIVE on the *CSD* dataset in order to perturb all categorical non-class attributes. Two non-class categorical attributes *Parents' Country* and *Favourite City* are totally uncorrelated with all other attributes and each of them has big domain size. Hence, any random perturbation to them would not affect the data quality of the perturbed dataset. Therefore, we apply DETECTIVE to perturb only other non-class attributes and examine the data quality of the perturbed dataset. We first apply DETECTIVE on the original *CSD* dataset in order to perturb categorical non-class attribute *Country of Origin*. We subsequently apply DETECTIVE two more times on the original dataset in order to perturb attributes *Car Make*, and *Profession*. Finally, we create a total perturbed dataset D_p (by combining these perturbed datasets) where attributes *Country of Origin*, *Car Make*, and *Profession* are perturbed. The rest of the attributes are unperturbed. We create decision trees $T_p(country)$, $T_p(car)$, $T_p(profession)$, and $T_p(status)$ (from the dataset D_p) that consider attribute *Country of Origin*, *Car Make*, *Profession*, and *Status* as class attribute, respectively. Subsequently, from the original dataset we build another set of decision trees $T_o(country)$, $T_o(car)$, $T_o(profession)$, and $T_o(status)$, that consider *Country of Origin*, *Car Make*, *Profession*, and *Status* as class attribute, respectively. We then compare the perturbed trees (trees obtained from perturbed datasets) with the corresponding original trees. For example, we compare each $T_p(car)$ with the $T_o(car)$. This comparison indicates the data quality of the perturbed dataset.

We run this experiments three times and produce three total perturbed datasets. Out of these three experiments two times we produce decision trees $T_p(car)$ which are exactly the same as the decision tree $T_o(car)$. Moreover, numbers of misclassified cases in these perturbed datasets are not significantly greater than the number of misclassified cases of the original dataset. Fig. 4.2 shows the tree applied to one of the two perturbed datasets while Fig. 4.1 shows the tree applied to the original dataset. Out of three experiments once we come up with a decision tree $T_p(car)$ which is slightly different to the decision tree $T_o(car)$. However, this $T_p(car)$ is just a pruned form of the decision tree $T_o(car)$ where logic rules for most of the records are same [Fig. 4.3]. Logic rules for other records are just pruned but very similar.

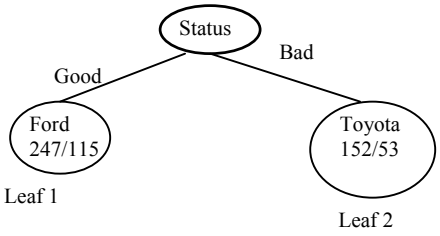

Fig. 4.3 Another decision tree built from a total perturbed *CSD* dataset. The tree considers attribute *Car Make* as class attribute.

Similarities of other perturbed trees with their corresponding original trees are also very high. We compare similarity of each perturbed tree with the similarity of corresponding original tree and present the result in Table 4.2. Our experimental results are very encouraging. Therefore, DETECTIVE appears to be successful in maintaining the pattern while providing privacy by data perturbation.

	Exactly Same	Very Similar	Similar	Dissimilar
$T_p(country)$	2	-	1	-
$T_p(car)$	2	1	-	-
$T_p(profession)$	3	-	-	-
$T_p(status)$	1	2	-	-

Table 4.2. Similarities of perturbed trees with corresponding original trees.

V. CONCLUSION

DETECTIVE introduces a novel concept of attribute specific horizontal partitioning, which is very useful for adding noised to categorical datasets. Additionally, it uses only relevant attributes instead of all the attributes. DETECTIVE is applicable to dataset with both numerical and categorical attributes. Our initial experiments indicate that DETECTIVE provides privacy preserving classification while preserving the patterns in the dataset. Above all, it is very simple and efficient. Our future work will include introduction of a new methods for quantifying similarity among categorical values and extensive experimental comparison of DETECTIVE and other existing clustering techniques.

VI. REFERENCES

[1] R.Agrawal and R. Srikant, "Privacy-preserving Data Mining", in *Proceedings of the ACM SIGMOD Conference on Management of Data*, Dallas, Tx, May14-19,2000.

[2] At Wal-Mart, World's Largest Reatail Data Warehouse Gets Even Larger, E. Schuman, Oct 13, 2004 (available from

www.eweek.com/article2/0,1759,1675960,00.asp) – visited on 29.03.05.

[3] D.Barbara, Y.Li and J.Couto, "COOLCAT: An entropy based algorithm for categorical clustering", in *Proc. of ACM Int. Conf. on Information and Knowledge Management,* 2002.

[4] K.T.Chuang and M.S.Chen, "Clustering Categorical Data by Utilizing the Correlated-Force Ensemble", in *Proc. of the 4th SIAM Int. Conf. on Data Mining (SDM 04)*, April 22-24,2004.

[5] Community Attitudes Towards Privacy 2004, A Survey prepared for The Office of the Federal Privacy Commissioner, Australia, Survey prepared by Roy Morgan Research, (available from www.privacy.gov.au/publications/rcommunity/index.html), June 18, 2004 – visited on 29.03.05.

[6] Data Mining: What is Data Mining?, J. Frand's web page at UCLA page (available from www.anderson.ucla.edu/faculty/json.frand/teacher/technologies/palace/datamining.htm) – visited on 29.03.05.

[7] V.Estivil-Castro and L.Brankovic, "Data Swapping: Balancing Privacy Against Precision in Mining for Logic Rules", in *Proceedings of the Data Warehousing and Knowledge Discovery (DaWaK 99)*, editors M.Mohania and A.M.Tjoa, 1999, 389-398.

[8] V.Ganti, J.Gehrke and R.Ramakrishnan, "CACTUS – Clustering Categorical Data Using Summaries", in *Proceedings of ACM SIGKDD,* 1999.

[9] S.Guha, R.Rastogi and K.Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes", in *Proceedings of the 15th Int. Conf. on Data Engineering,* March23-26, 1999, Sydney, Australia, p. 512.

[10] J. Han, and M. Kamber, "*Data Mining Concepts and Techniques*", Cerra, D. D., San Fransisco: 2001.

[11] M.Z.Islam and L. Brankovic, "Noise Addition for Protecting Privacy in Data Mining," in *Proceedings of the 6th Engineering Mathematics and Applications Conference (EMAC 2003),* 2003, Sydney, Australia, 85-90.

[12] M.Z.Islam, P.M.Barnaghi and L. Brankovic, "Measuring Data Quality: Predictive Accuracy vs. Similarity of Decision Trees", in *Proceedings of the 6th International Conference on Computer & Information Technology (ICCIT 2003)*, 2003, Dhaka, Bangladesh, Vol. 2, 457-462.

[13] M.Z.Islam and L. Brankovic, "A Framework for Privacy Preserving Classification in Data Mining", in *Proceedings of Australasian Workshop on Data Mining and Web Intelligence (DMWI 2004)*, Dunedin, New Zealand, CRPIT, **32**, J. Hogan, P. Montague, M. Purvis and C. Steketee, Eds., *Australasian Computer Science Communications*, 163-168.

[14] R.-H. Li, "Instability of Decision Tree Classification Algorithms", Ph.D. Thesis, University of Illinois at Urbana-Champaign, 2001.

[15] Wal-Mart Is Making Its Huge Data Warehouse Huger, at Teradata webpage, (available from www.teradata.com/t/page/ 129223/) – visited on 29.03.05.