Privacy Preservation in Data Mining Through Noise Addition

Md Zahidul Islam

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

THE UNIVERSITY OF
NEWCASTLE
AUSTRALIA

School of Electrical Engineering and Computer Science
University of Newcastle
Callaghan
New South Wales 2308
Australia

November 2007

Certificate of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree at any other University or Institution.

(Signed) ______________________

Md Zahidul Islam

Acknowledgements

I would like to thank my supervisor A/Prof. Ljiljana Brankovic who is something more than just a supervisor to all her students. Whenever I was in a trouble she was there with her genuine suggestions and dependable directions. She introduced this research area to me. If I have learnt anything on how to do research then it is due to her wise supervision. She always led us to be independent researchers having high ethics and moral values.

I would also like to thank my co-supervisor Professor A.S.M. Sajeev for his support, encouragement and wisdom. I am also grateful to Dr Regina Berretta, Dr Michael Hannaford, Dr Alexandre Mendes, Professor Mahbub Hassan, Professor M. F. Rahman, Professor Mirka Miller and Professor Elizabeth Chang for their support and encouragement.

I would give my special thanks to my friends Helen, Mousa, Mouris and Tanya for their enormous moral support throughout my study. My thanks also to all Faculty and Staff members of the School of Electrical Engineering and Computer Science and all postgraduate students of the school during my study for being so kind and friendly to me.

Last but not least, I would like to thank my wife Moonmoon for her patience, care, support, trust, love and encouragement. My special thanks to my children Abdellah, Saifullah and Mahir for their love and support. I would like to thank my parents Harun and Nilu, my father in law, mother in law, sister and brother in law for their encouragement. They have been a very supportive family all the way.

This thesis is gratefully dedicated to

My Family:

Moonmoon, my wife

Abdellah, **Saifullah** and **Mahir**, my sons

My Parents, **My Sister**, **My Father in law**, **My Mother in law**

and **All Relatives**

for their patience, their unwavering support and their faith.

Say: "If the ocean were ink (wherewith to write out) the words of my Lord. Sooner would the ocean be exhausted than would the words of my Lord, even if we added another ocean like it, for its aid." (Qur'an 18:109)

List of publications arising from this thesis

1. M. Z. Islam, and L. Brankovic, Privacy Preserving Data Mining: A Framework for Noise Addition to all Numerical and Categorical Attributes, In *Data Mining and Knowledge Discovery.* (In Preparation)

2. M. Z. Islam, and L. Brankovic, Privacy Preserving Data Mining: Noise Addition to Categorical Values Using a Novel Clustering Technique, In *IEEE Transactions on Industrial Informatics*, 2007. (Submitted on the 3rd September, 2007)

3. L. Brankovic, M. Z. Islam and H. Giggins, Privacy-Preserving Data Mining, Security, Privacy and Trust in Modern Data Management, Springer, Editors Milan Petkovic and Willem Jonker, ISBN: 978-3-540-69860-9, Chapter 11, 151-166, 2007.

4. M. Z. Islam, and L. Brankovic, DETECTIVE: A Decision Tree Based Categorical Value Clustering and Perturbation Technique in Privacy Preserving Data Mining, In *Proc. of the 3rd International IEEE Conference on Industrial Informatics*, Perth, Australia, (2005).

5. M. Z. Islam, and L. Brankovic, A Framework for Privacy Preserving Classification in Data Mining, In *Proc. of Australian Workshop on Data Mining and Web Intelligence (DMWI2004)*, Dunedin, New Zealand, CRPIT, **32**, J. Hogan, P. Montague, M. Purvis and C. Steketee, Eds., *Australian Computer Science Communications*, (2004) 163-168.

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

7. M. Z. Islam, and L. Brankovic, Noise Addition for Protecting Privacy in Data Mining, In *Proc. of the 6th Engineering Mathematics and Applications Conference (EMAC*

2003), Sydney, Australia, (2003) 457-462.

List of other publications during the candidature

1. M. Alfalayleh, L. Brankovic, H. Giggins, and M. Z. Islam, Towards the Graceful Tree Conjecture: A Survey, In *Proc. of the 15th Australasian Workshop on Combinatorial Algorithms (AWOCA 2004)*, Ballina, Australia, (2004).

Abstract

Due to advances in information processing technology and storage capacity, nowadays huge amount of data is being collected for various data analyses. Data mining techniques, such as classification, are often applied on these data to extract hidden information. During the whole process of data mining the data get exposed to several parties and such an exposure potentially leads to breaches of individual privacy.

This thesis presents a comprehensive noise addition technique for protecting individual privacy in a data set used for classification, while maintaining the data quality. We add noise to all attributes, both numerical and categorical, and both to class and non-class, in such a way so that the original patterns are preserved in a perturbed data set. Our technique is also capable of incorporating previously proposed noise addition techniques that maintain the statistical parameters of the data set, including correlations among attributes. Thus the perturbed data set may be used not only for classification but also for statistical analysis.

Our proposal has two main advantages. Firstly, as also suggested by our experimental results the perturbed data set maintains the same or very similar patterns as the original data set, as well as the correlations among attributes. While there are some noise addition techniques that maintain the statistical parameters of the data set, to the best of our knowledge this is the first comprehensive technique that preserves the patterns and thus removes the so called Data Mining Bias from the perturbed data set.

Secondly, re-identification of the original records directly depends on the amount of noise added, and in general can be made arbitrarily hard, while still preserving the original patterns in the data set. The only exception to this is the case when an intruder knows enough about the record to learn the confidential class value by applying the classifier. However, this is always possible, even when the original record has not been used in the training data set. In other words, providing that enough noise is added, our technique makes the records from the training set as safe as any other previously unseen records of the same kind.

In addition to the above contribution, this thesis also explores the suitability of prediction accuracy as a sole indicator of data quality, and proposes technique for clustering both categorical values and records containing such values.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the advances in information processing technology and the storage capacity, modern organisations collect a huge amount of data. For extracting hidden and previously unknown information from such huge data sets the organisations rely on various data mining techniques. During the whole process of data mining these data often get exposed to several parties. If such a party has enough supplementary knowledge about an individual having a record in the data set, then the party can re-identify the record. Thus sensitive information stored about the individual can potentially be disclosed resulting in a breach of individual privacy. Therefore, we need techniques for protecting individual privacy while allowing data mining.

Many noise addition techniques for protecting privacy have been designed for statistical databases [111, 102, 68, 56, 99, 100, 57, 58, 73, 92, 74] but they do not take into account the requirements specific to data mining applications. Wilson and Rosen investigated the prediction accuracy of classifiers obtained from data sets perturbed by existing noise addition techniques for statistical databases [116]. They found that the classifiers built on the data sets perturbed by such techniques including *GADP* suffer from a lack of prediction accuracy on a testing data set. This suggests the existence of a Data Mining Bias related to the change of patterns of a data set. We argue that two attributes may not be correlated over the whole data set, but may have a high correlation within a horizontal segment, i.e., within a set of records. For example, the attributes *Age* and *Weight* may not be highly correlated in the whole data set. However, they are typically correlated within the horizontal segment where age is less than 10 years. Therefore, when a perturbation technique preserves the overall correlation between two attributes it may not preserve the correlation between the attributes within a horizontal segment. This relationship between two attributes within a horizontal segment can be seen as a pattern of the data set.

Therefore, we need data modification techniques designed for various data mining applications such as classification and clustering. There are some noise addition techniques specifically designed for data mining [28, 3, 26, 121, 66]. The techniques proposed by Agrawal et al. and Du et al. [3, 26] perturb a data set targeting to maintain only good prediction accuracy of the classifier obtained from the perturbed data set. Zhu and Liu proposed a technique [121] that supports privacy preserving density estimation. The technique proposed by Estivill-Castro and Brankovic [28] focuses on preserving original patterns in a perturbed data set. They perturb a data set in such a way so that the classifier (decision tree) obtained from a perturbed data set is similar to the classifier obtained from the original data set.

We consider a data set as a two dimensional table where the rows (records) correspond to individuals and the columns (attributes) correspond to the properties of those individuals. Out of these attributes one is the categorical class attribute representing a class or category of a record. We consider the class attribute as confidential. The non-class attributes of a data set can be numerical or categorical. The technique proposed by Estivill-Castro and Brankovic [28] perturbs only the class attribute.

This thesis presents a collection of techniques that add noise to all attributes (including categorical and numerical non-class attributes, and the categorical class attribute), in such a way that the patterns from the original data set are also maintained in perturbed data sets. Our experimental results suggest that the decision trees obtained from the original data set and perturbed data sets are very similar in terms of logic rules and prediction accuracy. This recommends a high quality of perturbed data sets when used for classification. Additionally, when a relatively small amount of noise is added, the perturbed data sets are also suitable for other data mining tasks, such as clustering. In terms of security, noise added to all attributes makes record re-identification difficult, and thus protects individual privacy in a perturbed data set.

In **Chapter 2** we give a brief introduction to data mining and its applications. We also discuss the privacy issues related to data mining, and the growing public concern regarding this issue.

In **Chapter 3** we present a comprehensive background survey of existing techniques for privacy preserving data mining. We discuss the essence of these techniques and present a comparative study.

In **Chapter 4**, following the approach taken by Estivill-Castro and Brankovic [28], we present a technique for adding noise to a categorical class attribute in such a way so that the original patterns (logic rules) are not disturbed. Such a noise prevents an intruder from learning the confidential class values with certainty.

In **Chapter 5** we present techniques for adding noise to all non-class numerical attributes. The noise having zero mean and a normal distribution is added in such a way so that the perturbed value remains within an appropriate range defined by the original patterns of the data set. Therefore, a perturbed data set preserves the original patterns.

In **Chapter 6** we present a technique that adds noise to all non-class categorical attributes of a data set. Following the approaches taken by some existing techniques [40, 11, 61] we first cluster categorical values belonging to an attribute and then change them to other values belonging to the same cluster with some predefined probability. We use a novel clustering technique that has a few basic differences with existing clustering techniques. We also propose an extension of this technique to cluster the whole records rather than attribute values.

In **Chapter 7** we present a framework that combines the techniques for adding noise to all numerical and categorical attributes including the class attribute. Such noise addition prevents an accurate record re-identification by an intruder. Therefore, individual privacy in such perturbed data sets is expected to be very high. Our experimental results indicate that the perturbed data set preserves the original patterns and prediction accuracy very well.

In this chapter we also present an extended framework that incorporates an existing noise addition technique called *GADP* [73] along with the class attribute perturbation technique presented in Chapter 4. *GADP* perturbs a data set while preserving the original correlations among the attributes. Our extended framework uses *GADP* in such a way so that the perturbed values remain within appropriate ranges defined by the original patterns. Therefore, the extended framework preserves the original patterns along with the correlations. The extended framework can also use *C-GADP* [92] or *EGADP* [74] instead of *GADP* to accommodate small data sets that do not have multivariate normal distribution. We finally present experimental results that suggest the effectiveness of our framework and extended framework.

In **Chapter 8** we present a novel technique for measuring disclosure risk of a perturbed data set. The technique uses an entropy based approach. We discuss our noise addition method and show that we can achieve arbitrarily high security by adding required level of noise to all non-influential attributes, while still preserving the patterns in the data set.

In **Chapter 9** we explore the suitability of prediction accuracy as a sole indicator of data quality. Typically data quality is evaluated by just the prediction accuracy of the classifier obtained from the data set [60, 69, 116]. However, in Chapter 7 we evaluated data quality of a perturbed data set through a few quality indicators such as the similarity of decision trees obtained from the original and the perturbed data set, prediction accuracy of a classifier on the underlying data set and on a testing data set and the similarity of correlation matrices belonging to the original and the perturbed data set.

In **Chapter 10** we present concluding remarks and future research directions.

Chapter 2

Data Mining

In this chapter we give a brief introduction to data mining, its application and main techniques. We also discuss privacy issues arising in data mining, and the related growing public concern.

2.1 Introduction to Data Mining

2.1.1 Definition

Since the introduction of relational databases a series of advanced data models have been developed including extended-relational, object-oriented, object-relational and deductive databases. Today we have application oriented database systems, heterogeneous database systems and Internet-based global information systems such as World Wide Web (WWW). At the same time, there has been a remarkable progress in computer hardware technology. This has led to a huge supply of powerful and affordable computers, data collection equipments and storage media in the market. Due to this development of information processing technology and storage capacity, huge amount of data is being collected on a regular basis by various organizations and companies. Data mining techniques are used to extract invaluable knowledge from this huge amount of data.

It has been pointed out that when gold is mined from rocks or sand we call it gold mining instead of rock mining or sand mining [47]. Therefore, data mining could alternatively be called knowledge mining. The reason why it is called data mining instead of knowledge mining is perhaps to emphasise the fact that it is applied on huge amount of data. It would not be possible to fetch fruitful knowledge from such gigantic data sets without the help of data mining.

Data mining can be seen as a process for extracting **hidden** and **valid** knowledge from **huge databases** [13]. The highlighted words point to the essential characteristics of data mining. These characteristics are further clarified as follows.

Data mining extracts knowledge which was previously unknown [14]. This means that the extracted knowledge could not even be hypothesized in advance. The more unexpected the knowledge generally the more interesting it is. There is no benefit of mining a big data set to extract the knowledge which is very obvious. For example, regression analysis was used at some point in time in the Vietnam war to predict the possible mortar attacks [44]. After extensive analysis of huge amount of data a conclusion was made that there was a period during which increased level of mortar attacks could be predicted with reasonable certainty. This period was during the new moon. Such analysis and discovery of knowledge was absolutely useless since, everyone already knew that mortar attacks were much more likely when it was dark.

The extracted knowledge needs to be valid. Supermarket chains usually have huge number of items and transactions in their transactional databases. Some interesting associations between few items can be discovered if someone looks into the database carefully. However, such an association may not be a valid one in the sense that the items may not appear together in sufficient number of transactions. Moreover, the appearance of one of the items may not necessarily suggest the appearance of other items with sufficient certainty in the whole transactional database.

Extraction of knowledge from a data set having a small number of records is not considered as data mining. Such knowledge can be obtained manually through an observation and use of traditional data analysis. Extraction of knowledge from a huge data set is a basic characteristic of data mining.

2.1.2 Comparison with Traditional Data Analyses

Often there is confusion among the end users about the differences between a traditional data analysis and data mining. A basic difference is that, unlike traditional data analyses, data mining does not require predefined assumptions. A difference between data mining and traditional data analysis is illustrated with an example as follows [47, 44, 114, 52].

A regional manager of a chain of electronics stores may use traditional data analysis tools to investigate the sales of air conditioners in different quarters of the year. The regional manager can also use traditional data analysis tools to analyse the relationship between the sales of computers and printers in different stores of his region. Both of these scenarios have one thing in common: an assumption. In the first scenario the regional manager assumes that the sales volume of air conditioners depends on the weather and temperature. In the second scenario he assumes a relation between the sales of computers and printers. An end user needs some sort of assumption to formulate a query in traditional data analysis. Conversely, data mining lifts such barriers and allows end users to find answers to questions that can not be handled by traditional data analysis. For example, the regional manager could look for answers to questions such as: Why the sales of computers in few stores are not as high as the sales of computers in other stores? How can I increase the overall sales in all stores?

Among all traditional data analyses, statistical analysis is the most similar one to data mining. Many of the data mining tasks, such as building predictive models, and discovering associations, could also be done through statistical analysis. An advantage of data mining is its assumption free approach. Statistical analysis still needs some predefined hypothesis. Additionally, statistical analysis is usually restricted to only numerical attributes while data mining can handle both numerical and categorical attributes. Moreover, data mining techniques are generally easy to use. A combination of data mining and statistical techniques can produce a more efficient data analysis.

2.1.3 Data Mining Steps

Some consider data mining as a synonym for another popular term Knowledge Discovery in Database (KDD), while some believe data mining is an integral part of KDD [47]. Essential steps of KDD include Data Cleaning, Data Integration, Data Selection, Data Mining, Pattern Evaluation and Knowledge Presentation [47, 44]. Each of these steps are also briefly discussed as follows.

Data Cleaning

Data cleaning generally refers to the removal of natural noise and inconsistent data from the database [47, 44, 52]. Words and numbers could be misspelled and entered erroneously in a database due to various reasons including typographical errors. Detection and rectification of such errors is an essential part of KDD. Sometimes databases contain missing values which may cause ineffective data mining if they are left unresolved. Missing values are either replaced by the most likely value or deleted along with the whole record. Another data cleaning issue arises from inconsistent data entries. An attribute "Soft Drink" may have values such as "Pepsi", "Pepsi Cola" and "Cola" which necessarily refer to the same drink [44]. They need to be made consistent before the application of a data mining technique.

Data Integration

Data integration (also known as data transformation) is the process of combining two or more data sources into a uniform data set. Different data sources may use different data models such as relational and object-oriented database model [47, 44, 52]. A two dimensional data set is created from these various sources. From a relational data model a two dimensional data set can be created in many ways including merging the relational tables in a view and exporting the data into a flat file.

Sometimes a particular attribute can be called differently in different data sources. For example, one data source may name an attribute as "Income", while another source may name the same attribute as "Salary". Same values can also be stored differently in different sources. For example, in one source values of the attribute "Sex" can be "Male" and "Female", whereas they can be "M" and "F" in another source. These anomalies are resolved in the data integration phase.

Sometimes various data sources may store different information such as "Orders" and "Customers" data. These data are also integrated in data integration phase. Data cleaning and integration are sometimes together referred as data preparation [47, 44, 52].

Data Selection

In order to perform a particular data mining task all relevant attributes are selected from the warehouse data set. This new data set comprising of these selected attributes are used for data mining.

Data Mining

Data mining is the essential process of KDD which extracts previously unknown patterns and trends from a huge data set without making any predefined hypothesis.

Pattern Evaluation and Knowledge Presentation

Some extracted patterns are obvious and unappealing, for example an increased likelyhood of mortar attacks when dark during the Vietnam war. However, some other patterns may be counterintuitive, interesting and useful. Discovered patterns are therefore evaluated before they are presented to an end user in a user understandable form. Various visualization and knowledge representation techniques are used to prepare a meaningful presentation.

2.2 Data Mining Tasks

There are some tasks that make use of data mining techniques, although they themselves are not data mining. These tasks are often mistakenly considered as data mining, perhaps due to their close link to it. An example of such tasks is the detection of a fraudulent credit card use [114], which we briefly outline as follows.

Credit card companies apply data mining techniques on data sets having large number of previous credit card transactions. Some of these transactions may be known as fraudulent which the credit card company uses to learn the patterns of fraudulent credit card uses. An example of such patterns could be a purchase of gasoline for a small amount, followed by another gasoline purchase for a bigger amount and other series of purchases for large amounts [114]. What actually happens is the following: when a card is stolen it is usually tested through a small purchase of gasoline at a pay-at-the-pump service station and then used for bigger purchases if the card is found still active. The credit card company constantly monitors each transaction of a credit card and matches the transaction trends with the discovered patterns. This matching is generally done through some online pattern matching operations. Discovery of the patterns is data mining. However, the detection of a fraudulent card use through pattern matching is not data mining.

There are many data mining tasks, such as Classification, Association Rule Mining, Clustering, Outlier Analysis, Evolution Analysis, Characterization and Discrimination [47, 52]. We briefly discuss few of them as follows.

Classification and Prediction

A data set may have an attribute called “class attribute” which refers to the category of the records. For example, a patient data set may have a class attribute called “Diagnosis” along with several other non-class attributes that describe various properties and conditions of a patient. Class attribute is also known as labels. Records having class attribute values are known as labelled records [47].

Classification is the process of building a classifier from a set of pre-classified (labelled) records. It discovers a pattern (model) that explains the relationship between the class and the non-class attributes [47]. A classifier is then used to assign (predict) a class attribute value to new unlabeled records. Classifiers also help to analyze the data sets better. They are expressed in different ways such as decision trees, sets of rules.

One of the techniques for building decision trees is based on information gain [85]. This technique first calculates the entropy (uncertainty) in estimating the class attribute values in the whole data set. It then divides the whole data set into two parts, based on an attribute, where each part contains a subset of values of the attribute and the other part contains the set of remaining values of the attribute. The attribute value that sits in the border of the two sets of values is also known as the splitting point.

Due to the division of the data set, the uncertainty in estimating the class attribute value changes which depends on the distribution of the class values. For example, let us assume that the domain size of the class attribute of a data set is two. In an extreme case, if all records belonging to one division have one class value and all other records belonging to the other division have the other class value then the uncertainty gets reduced to zero resulting in the maximum information gain. The decision tree building algorithm picks the best splitting point, among all possible splitting points of all non-class attributes, that reduces the uncertainty the most. The best splitting attribute is the root node of a decision tree and the best splitting point is the label on the edges. A reader can consult Figure 2.1 for better understanding. The same approach is applied again on each division of the data set and this process continues until the termination condition is met, resulting in a decision tree classifier.

We next describe the use of decision trees in understanding/analyzing a data set and for classifying/predicting the class values of new unlabeled records. When a tree is used to classify an unlabeled record, the record is first tested in the root of the tree. Depending on

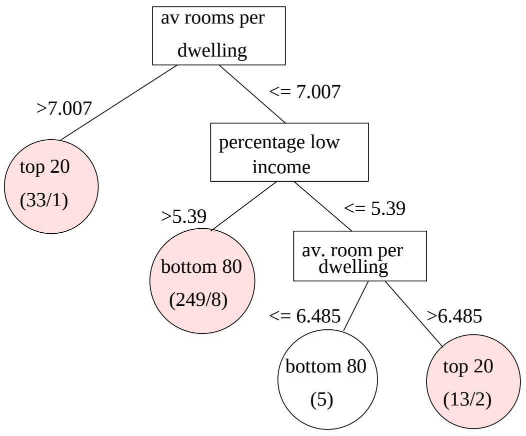

Figure 2.1: An example of a decision tree. Squares represent internal nodes, the unshaded circle represents homogeneous leaf where all records have the same class value and shaded circles represent heterogeneous leaves.

the outcome of the test an edge from the root is taken and subsequent test is performed in the next node. This procedure is repeated until the record arrives in one of the leaves of the tree and the leaf class is assigned to the record. Classifiers are also used in analyzing the patterns in a data set through a close observation of the classification rules. Classifiers are sometimes used to predict the possible value of a missing attribute value of a data set.

If the attribute tested in a node is numerical then typically there are two edges from the node. One of the edges is labelled "$> c$" while the other edge is labelled "$<= c$", where c is a constant from the domain of that attribute. If, on the other hand, the attribute is categorical then there are typically a few edges from the node, each labelled by a category from the attribute domain. Each leaf of the tree has a class associated with it. In homogeneous leaves the leaf class appears in all the records from the training set (the data set from which the tree has been built) that belong to that leaf. In heterogeneous leaves majority of the records from the training set belong to the leaf class and only a few belong to another class.

An example of a decision tree is shown in Figure 2.1. The tree was built on the Boston Housing Price dataset which is available from the UCI Machine Learning Repository [77]. Each record corresponds to a suburb and attributes include average number of rooms per dwelling and percentage lower income earners. The class refers to the median house price in a suburb and has two values, top 20% and bottom 80%. The root node tests the attribute av rooms per dwelling. This is a numerical attribute and the left edge from the root denotes the values greater than 7.007 while the right edge denotes values less than or equal to 7.007.

If the left edge is followed we arrive in a heterogeneous leaf containing 33 cases/records from the training set where all but one belong to the top 20 %.

Association Rules Mining

In order to explain association rule mining we consider a set of transactions (records) where each transaction consists of a list of items, which are categorical values, such as {milk, coke, bread} [47]. Transactions may significantly vary in size and some transactions may have only few items, whereas others may have many items. Another commonly used form of the transactional data set is market basket data set where rows/records are transactions of the same size, which is equal to the size of the set of all possible items. Each transaction is a series of 0s and 1s, where a 0 for an item represents the absence of the item in the transaction and a 1 represents the presence of the item.

Primary objectives of association rule mining are to obtain frequent item sets, and association rules. If a set of items appears in a number of transactions which is more than a user defined threshold than the set is called a frequent item set, with respect to the threshold. However, if two sets of items are associated in such a way that the appearance of one set of items in a transaction makes the appearance of the other set in the same transaction highly expected then this is called an association rule.

An association rule can be represented as $X \Rightarrow Y$, where X and Y are mutually exclusive subsets of items and $X, Y \subset I$, where I is the set of all items of the data set. A rule $X \Rightarrow Y$ has a support factor s if at least $s\%$ of the transactions of the whole data set satisfies the rule. A rule $X \Rightarrow Y$ is said to have a confidence factor c if $c\%$ of the transactions that satisfy X also satisfy Y.

An example of association rule is "$computer \Rightarrow software$ $[1\%, 50\%]$" which says that if a transaction contains "*computer*" then there is a 50% chance that it will also contain "$software$". Additionally, 1% of all of the transactions contain both of the items [47]. An association rule having high support and confidence is generally considered significant and interesting. Significant rules are used to learn the buying patterns, to plan marketing strategies and so on.

Association rule mining generally takes a transactional data set and other threshold values (such as threshold confidence and threshold support) as input, and generates a set of interesting association rules as the output. It can also be applied on many other types of

data sets such as a customer data set [47], and can discover association rules of the form *"age(20,...29) and income(>55)* $\Rightarrow$ *buys(Sports Car),*
[*support* = *2%*, *confidence* = *60%*]".

Clustering

Clustering is the process of arranging similar records in groups so that the records belonging to the same cluster have high similarity, while records belonging to different clusters have high dissimilarity [47]. Unlike classification, clustering usually analyzes unlabeled records. In many cases, a data set may not have a class attribute when it is initially collected. Class labels are generally assigned to these records based on the clusters. Typical applications of clustering include discovery of distinct customer groups, categorization of genes with similar functionality and identification of areas of similar land use [47].

There exists a large number of clustering methods including partitioning, hierarchical, density based, grid based and model based methods, shown in Figure 2.2. We briefly discuss these methods as follows [47].

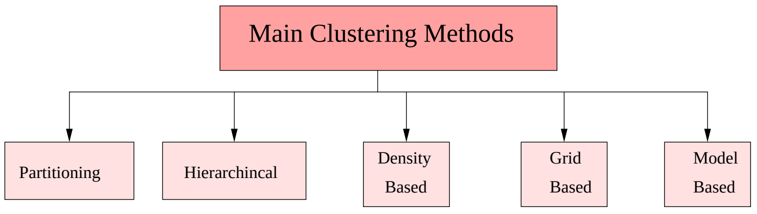

Figure 2.2: Main Clustering Methods.

A partitioning method generally divides the records of a data set into k non-empty and mutually exclusive partitions, where k is a user defined number. Note that $k \leq n$, where n is the number of the records. The method then uses an iterative relocation in order to improve the quality of the partitions/clusters by grouping similar records in a cluster and dissimilar records in different clusters. The two common heuristics used in this method are k-means and k-medoids.

A hierarchical method can be further divided into two types, agglomerative and divisive. An agglomerative hierarchical method first considers each single data object/record as a separate cluster. Based on some similarity criteria it then merges the two most similar records or groups of records in each successive iteration until it fulfills a termination condition or all records are merged into one single cluster. On the other hand, the divisive hierarchical clustering method starts with all records in a single cluster. In each iteration, it splits a cluster into two clusters in order to improve the criteria that measures the goodness of the overall clustering. Finally, the method stops when a termination condition is met or each record is separated into a cluster.

A density based clustering forms clusters of dense regions where a high number of records are located. It initially selects a core record that has large number of neighbor records. The core record and all its neighbor records are included in a cluster. If a record r among these neighbors is itself a core, then all neighbors of r are also added in the cluster. The process terminates when there is no record left that can be added to a cluster. This clustering technique is also used to filter out noise from a data set.

A grid based method performs all clustering operations on a grid like structure obtained by quantizing the data space into a finite number of cells. The main advantage is a faster processing speed which mainly depends on the number of cells.

Unlike conventional clustering, a model based clustering attempts to find a characteristic description of each cluster, in addition to just clustering the unlabeled records. Each cluster represents a concept or class. Some model based clustering technique such as COBWEB generates a hierarchical clustering in the form of a classification tree. Each node of the classification tree refers to a concept in a hierarchical system. Each node also presents a description of the concept that summarizes the records classified under the node.

2.3 Applications of Data Mining

2.3.1 Usefulness in General

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. Business organizations collect data about the consumers for marketing purposes and improving business strategies, medical organizations collect medical records for better treatment and medical research, and national security agencies maintain criminal records for security purposes. 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 [14, 95, 112, 113]. According to Teradata, Wal-Mart has plans to expand its huge warehouse to even huger, allegedly to a capacity of 500 tera bytes [95]. Wal-Mart allows more than 3,500 suppliers to access its huge data set and perform various data analyzes.

For successful analyzes of these huge sized data sets various data mining techniques are widely used by the organizations all over the world. For example, Wal-Mart uses its data set for trend analysis [44]. In modern days organizations are extremely dependent on data mining in their every day activities. Data mining techniques 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.

2.3.2 Some Applications of Data Mining Techniques

There is a wide range of data mining applications. A few of them are discussed as follows.

Medical Data Analysis

Generally, medical data sets contain wide variety of bio-medical data which are distributed among parties. Examples of such databases include genome and proteome databases. Various data mining tasks such as data cleaning, data preprocessing and semantic integration can be used for the construction of warehouse and useful analysis of these medical databases [46].

Data mining techniques can be used to analyse gene sequences in order to find genetic factors of a disease and the mechanism that protect the body from the disease. A disease can be caused by a disorder in a single gene, however in most cases disorder in a combination of genes are responsible for a certain disease. Data mining techniques can be used to indicate such a combination of genes in a target sample. Data sets having patient records can also be analyzed through data mining for various other purposes such as prediction of diseases for new patients. Moreover, data mining is also used for the composition of drugs tailored towards individual's genetic structure [103].

Direct Marketing

In direct marketing approach a company delivers its promotional material such as leaflets, catalogs, and brochures directly to potential consumers through direct mail, telephone marketing, door to door selling or other direct means. It is crucial to relatively precisely identify potential consumers in order to save marketing expenditure of the company. Data mining techniques are widely used for identifying potential consumers by many companies and organisations including People's Bank, Reader's Digest, the Washington Post and Equifax [44].

Trend Analysis

Trend analysis is generally used in stock market studies where the essential task is the so called bull and bear trend analysis. A bull market is the situation where prices rise consistently for a prolonged period of time, whereas a bear market is the opposite situation [115].

Financial institutions require to realize and predict customer deposit and withdrawal pattern. Supermarket chains need to identify customers' buying trends and association rules (i.e. which items are likely to be sold together). Wal-Mart is one of the many organizations that uses data mining for trend analysis [44].

Fraud Detection

Fraudulent credit card uses cost the industry over a billion dollars a year [44, 36]. Almost all financial institutions, such as MasterCard, Visa, and Citibank, use data mining techniques to discover fraudulent credit card use patterns [44]. The use of data mining techniques has already started to reduce the losses. The Guardian (September 9, 2004) published that the loss due to credit card fraud reduced by more than 5% in Great Britain in 2003 [36]. Similarly mobile phone frauds are also very common all over the world. According to Ericsson more than 15,000 mobile phones are stolen just in Britain every month [36]. Data mining techniques are also used to prevent fraudulent users from stealing mobile phones and leaving bills unpaid.

Plagiarism Detection

Assignments submitted by the students can be characterized by several attributes. For example, the attributes for a programming assignment can be run time for a program, number of integer variables used, number of instructions generated, and so on. Based on the attribute values the submissions are analyzed through clustering, which groups similar submissions together [22]. Submissions that are clustered together can be suspected for plagiarism.

2.4 Privacy Issues Related to Data Mining

Every day we are leaving dozens of electronic trails through various activities such as using credit cards, swapping security cards, talking over phones and using emails [15]. Ideally, the data should be collected with the consent of the data subjects. The collectors should provide some assurance that the individual privacy will be protected. However, the secondary use of collected data is also very common. Secondary use is any use for which data were not collected initially. Additionally, it is a common practice that organizations sell the collected data to other organizations, which use these data for their own purposes.

Nowadays, data mining is a widely accepted technique for huge range of organizations. Organizations are extremely dependent on data mining in their every day activities. The paybacks are well acknowledged and can hardly be overestimated. During the whole process of data mining (from collection of data to discovery of knowledge) these data, which typically contain sensitive individual information such as medical and financial information, often get exposed to several parties including collectors, owners, users and miners. Disclosure of such sensitive information can cause a breach of individual privacy. For example, the detailed credit card record of an individual can expose the private life style with sufficient accuracy. Private information can also be disclosed by linking multiple databases belonging to giant data warehouses [34] and accessing web data [101].

An intruder or malicious data miner can learn sensitive attribute values such as disease type (e.g. HIV positive), and income (e.g. AUD 82,000) of a certain individual, through re-identification of the record from an exposed data set. We note that the removal of the names and other identifiers (such as driver license number and social security number) may not guarantee the confidentiality of individual records, since a particular record can often be uniquely identified from the combination of other attributes. Therefore, it is not difficult for an intruder to be able to re-identify a record from a data set if he/she has enough supplementary knowledge about an individual. It is also not unlikely for an intruder to have sufficient supplementary knowledge, such as ethnic background, religion, marital status and number of children of the individual.

Public Awareness

There is a growing anxiety about delicate personal information being open to potential misuses. This is not necessarily limited to data as sensitive as medical and genetic records. Other personal information, although not as sensitive as health records, can also be considered to be confidential and vulnerable to malicious exploitation. For example, credit card records, buying patterns, books and CDs borrowed, and phone calls made by an individual can be used to monitor his/her personal habits.

Public concern is mainly caused by the so-called secondary use of personal information without the consent of the subject. In other words, consumers feel strongly that their personal information should not be sold to other organizations without their prior consent. Recent surveys reflect this as discussed below.

The IBM Multinational Consumer Privacy Survey performed in 1999 in Germany, USA and UK illustrates public concern about privacy [87]. Most respondents (80%) feel that "consumers have lost all control over how personal information is collected and used by companies". The majority of respondents (94%) are concerned about the possible misuse of their personal information. This survey also shows that, when it comes to the confidence that their personal information is properly handled, consumers have most trust in health care providers and banks and the least trust in credit card agencies and internet companies.

A Harris Poll survey illustrates the growing public awareness and apprehension regarding their privacy, from survey results obtained in 1999, 2000, 2001 and 2003 [98]. The public awareness regarding their privacy is shown in Table 2.1.

In 2004, the Office of the Federal Privacy Commissioner, Australia, engaged Roy Morgan Research to investigate community attitude towards privacy [88]. According to the survey, 81% of the respondents believe that "customer details held by commercial organizations 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

-	1999	2000	2001	2003
Concerned	78%	88%	92%	90%
Unconcerned	22%	12%	8%	10%

Table 2.1: Harris Poll Survey: Privacy Consciousness of Adults [98]

not know as a privacy invasion. Secondary use of personal information by the collecting organization is considered as a privacy invasion by 93% of the respondents. Respondents were found most reluctant to disclose details about finances (41%) and income (10%).

Public awareness about privacy and lack of public trust in organizations may introduce additional complexity to data collection. For example, strong public concern may force governments and law enforcing agencies to introduce and implement new privacy protecting laws and regulations such as the US Executive Order (2000) that protects federal employees from being discriminated, on the basis of protected genetic information, for employment purposes [78]. It is not unlikely that stricter privacy laws will be introduced in the future. On the other hand, without such laws individuals may become hesitant to share their personal information resulting in additional difficulties in obtaining truthful information from individuals.

Both scenarios may make the data collection difficult and hence may deprive the organizations from the benefits of data mining resulting in inferior quality of services provided to the public. Such prospects equally concern collectors and owners of data, as well as researchers.

Privacy Preserving Data Mining

Due to the enormous benefits of data mining, yet high public concerns regarding individual privacy, the implementation of privacy preserving data mining techniques has become a demand of the moment. A privacy preserving data mining provides individual privacy while allowing extraction of useful knowledge from data.

There are several different methods that can be used to enable privacy preserving data mining. One particular class of such techniques modifies the collected data set before its release, in an attempt to protect individual records from being re-identified. An intruder even with supplementary knowledge, can not be certain about the correctness of a re-identification, when the data set has been modified. This class of privacy preserving techniques relies on the fact that the data sets used for data mining purposes do not necessarily need to contain 100% accurate data. In fact, that is almost never the case, due to the existence of natural noise in data sets. In the context of data mining it is important to maintain the patterns in the data set. Additionally, maintenance of statistical parameters, namely means, variances and covariances of attributes is important in the context of statistical databases.

High data quality and privacy/security are two important requirements that a good privacy preserving technique needs to satisfy. We need to evaluate the data quality and the degree of privacy of a perturbed data set. Data quality of a perturbed data set can be evaluated through a few quality indicators such as extent to which the original patterns are preserved, and maintenance of statistical parameters. There is no single agreed upon definition of privacy. Therefore, measuring privacy/security is a challenging task.

2.5 Conclusion

In this chapter we have given a brief introduction to data mining and its application. We have also discussed the privacy issues related to data mining and the growing public concern regarding their privacy. Due to the huge public concern we need privacy preserving data mining techniques. In the next chapter we present a background study on privacy preserving data mining.

Chapter 3

Privacy Preserving Data Mining - A Background Study

In this chapter we present a background study on techniques for privacy preserving data mining. In Section 3.1 a classification scheme and evaluation criteria is presented. In Sections 3.2 and 3.3 we give a comprehensive overview of the existing techniques. The fundamental approach and essence of each of these techniques are presented. Some comparative discussions about the techniques are also offered in order to present their strengths and weaknesses. In Section 3.4 we give a comparative study of several privacy preserving techniques in a tabular form. Finally, Section 3.5 presents a conclusion of the section.

3.1 Classification Scheme and Evaluation Criteria

A number of different techniques has been proposed for privacy preserving data mining. Each of these techniques is suitable for a particular scenario and objective. In this section we propose a classification scheme and evaluation criteria for these techniques. Our classification scheme and evaluation criteria builds upon but do not strictly follow the classification scheme and evaluation criteria proposed in [109].

Privacy preserving techniques can be classified based on the following characteristics.

- Data Distribution
- Data Type

- Privacy Definition
- Data Mining Scenario
- Data Mining Tasks
- Protection Methods

We illustrate these classification characteristics as follows.

Data Distribution

The data sets used for data mining can be either centralized or distributed. This does not refer to the physical location where data is stored, but to the availability/ownership of data. Centralized data set is owned by a single party. It is either available at the computation site or can be sent to the site. However, distributed data set is shared between two or more parties which do not necessarily trust each other with their private data, but are interested in mining their joint data. Data owned by each party is a portion of the total data set which is distributed among the parties. The data set can be heterogenous, i.e. vertically partitioned, where each party owns the same set of records but different subset of attributes. Alternatively, the data set can be homogenous, i.e. horizontally partitioned, where each party owns the same set of attributes but different subset of records. Figure 3.1 shows this classification of the data sets.

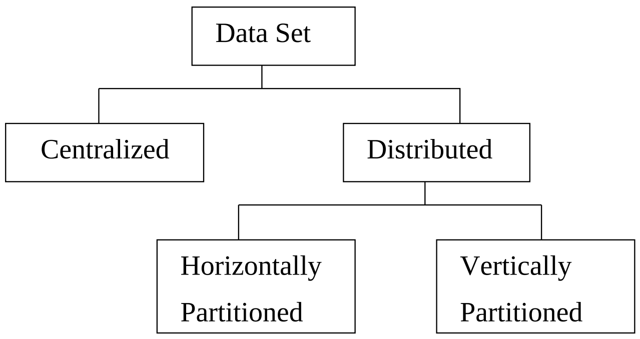

Figure 3.1: Classification of Data Sets Based on Distribution.

Centralized data are usually more complete than a portion of a distributed data, in the sense that they contain sufficient number of records and relevant attributes to serve the purpose of the data collection and mining. A data set having insufficient number of records and/or attributes can also be considered centralized in case of the unavailability of other data sets to share with. If such other data sets are available and the parties decide to combine their data sets under a single ownership then the combined data set is still centralized. However, mutual untrust and conflicts of interest usually discourage the parties from combining such data sets.

We next give examples of centralized and distributed data sets. Two health service providers such as hospitals may have portions of a horizontally partitioned data set. Mining any of these portions may not be as fruitful as mining the joint data set. In distributed data mining although the hospitals mine their joint data set, none of them discloses its data to the other hospital. In another scenario two organizations such as a taxation office and a social security office may have portions of a vertically partitioned data set.

Data Type

An attribute in a data set can be either categorical or numerical. Boolean data are a special case of categorical data, which can take only two possible values, 0 or 1. Categorical values lack natural ordering in them. For example, an attribute “Car Make” can have values such as “Toyota”, “Ford”, “Nissan”, and “Subaru”. There is no straightforward way of ordering these values. This fundamental difference between categorical and numerical values forces the privacy protection techniques to take different approaches for them.

Privacy Definition

The definition of privacy is context dependant. In some scenarios individual data values are private, whereas in other scenarios certain association or classification rules are private. For example, we consider a scenario where a health service provider releases its patient data set to facilitate research and general analyzes. They may consider sensitive attribute values belonging to an individual as private. A privacy protection technique should prevent a disclosure of such a sensitive attribute value. However, in another scenario two or more organizations decide to collaborate by releasing their data sets to each other for mining. They expect this collaboration to give them an advantage over the rest of their competitors who did not participate in the collaboration. Nevertheless, the collaborating organizations may not want to disclose some sensitive rules hidden in their corresponding data sets.

Therefore, in this scenario a privacy protection technique should prevent a disclosure of sensitive rules.

Data Mining Scenario

There are two major data mining scenarios. In the first scenario organizations release their data sets for data mining allowing unrestricted access to it. However, the individual privacy is protected in the released data sets usually by data modification. In the second scenario organizations do not release their data sets, but still allow data mining involving their data sets. Typically, cryptographic techniques are used for privacy preserving data mining in this scenario.

Data Mining Tasks

A data set contains various types of patterns. These patterns can be extracted through a variety of data mining tasks such as classification, clustering, association rule mining, outlier analysis, and evolution analysis [47]. In many occasions a user may need to perform different data mining tasks on a released data set in order to discover interesting patterns.

Ideally, a privacy preserving technique should maintain the data quality to support all possible data mining tasks and statistical analysis. However, it usually maintains the data quality to support only a group of data mining tasks. The privacy protection techniques can be classified based on the tasks which are supported by them.

Protection Methods

Privacy can be protected through different methods such as *Data Modification* and *Secure Multi-party Computation.* Privacy preserving techniques can be classified based on the protection methods used by them. This classification is shown in Figure 3.2.

Data Modification techniques modify a data set before releasing it to the users. Data is modified in such a way that the privacy is preserved in the released data set, whereas the data quality remains high enough to serve the purpose of the release. A data modification technique could be developed to protect the privacy of individuals, sensitive underlying patterns, or both. This class of techniques include *noise addition*, *data swapping*, *aggregation*, and *suppression.*

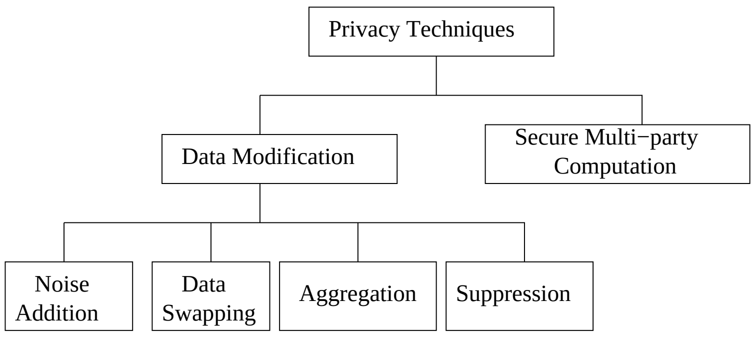

Figure 3.2: A Classification of Privacy Preserving Techniques.

Noise addition usually adds a random number (noise) to numerical attributes. This random number is generally drawn from a normal distribution with zero mean and a small standard deviation. Noise is added in a controlled way so as to maintain means, variances and co-variances of the attributes of a data set. However, noise addition to categorical attributes are not as straightforward as the noise addition to numerical attributes, due to the absence of natural ordering in categorical values. *Data swapping* interchanges the attribute values among different records. Similar attribute values are interchanged with higher probability. All original values are kept within the data set, just the positions are swapped. Note that in statistical database terminology, data swapping is often seen as a special case of noise addition. The reason for that is - swapping two numerical values can be seen as the addition of a number (the difference between the values) to the smaller value, and subtraction of the same number from the larger value. Therefore, data swapping results in the addition of noise having zero mean. Similar explanation can be given for swapping categorical values. *Aggregation* refers to both combining a few attribute values into one, or grouping a few records together and replacing them with a group representative. Finally, *Suppression* means replacing an attribute value in one or more records by a missing value.

Secure Multi-party Computation (SMC) techniques encrypts the data sets, while still allowing data mining tasks. Ideally, SMC techniques are not supposed to disclose any new information other than the final result of the computation to a participating party. These techniques are typically based on cryptographic protocols and are applied to distributed data sets. Parties involved in a distributed data mining encrypt their data and send to others. These encrypted data are used to compute the aggregate data, belonging to the joint data set, which is used for data mining.

A classification scheme helps us to find the group of techniques that is suitable for a scenario. Many such techniques for different scenarios have already been proposed. It is unlikely to have a single privacy preserving technique that outperforms all other existing techniques in all aspects. Each technique has its strength and weakness. Hence, a comprehensive evaluation of a privacy preserving technique is crucial. It is important to determine the evaluation criteria and related benchmarks. We discuss some evaluation criteria as follows.

1. *Versatility* refers to the ability of the technique to cater for various privacy requirements, types of data sets and data mining tasks. The more versatile a technique, the more useful it is. Versatility includes the following.

   - *Private: Data vs Patterns*

     A group of privacy preserving techniques considers sensitive attribute values as private, whereas another group considers sensitive patterns (such as some sensitive association rules) as private. A versatile technique would consider both individual data and some patterns as private.

   - *Data set: Centralized or Distributed (Vertical or Horizontal)*

     Usually privacy preserving techniques are suitable for either centralized or a distributed data sets. However, some versatile techniques are suitable for both types of data sets.

   - *Attributes: Numerical or Categorical (Boolean)*

     Many privacy preserving techniques can handle either numerical or categorical (considering boolean as a special case of categorical) data. However, there are many data sets, indeed, that have both types of data. Therefore, a versatile technique which is capable of managing both types of data can be a perfect choice for these data sets.

   - *Data Mining Task*

     A user initially may not know which patterns are interesting in a released data set. He may need to carry out several data mining tasks such as classification, and clustering on a released data set to discover interesting and useful information. Privacy protection techniques are used on the original data sets before they are released. Therefore, it is preferable that a privacy protection technique supports as many data mining tasks as possible. Otherwise, the user can only perform that data mining task which is supported by the privacy protection technique.

2. *Disclosure Risk*

   Disclosure risk refers to the possibility of sensitive information being inferred by a malicious data miner. It is inversely proportional to the level of security offered by the technique. Since the primary objective of the use of a privacy protection technique is to minimize the disclosure risk, the evaluation of the risk is essential. Evaluation of a disclosure risk is also a challenging task, since it depends on many factors including what is already known (the supplementary knowledge) to an intruder, and the nature of the technique. For example, in a technique that protects sensitive association rules, a disclosure risk might be measured by the percentage of the sensitive rules that can still be disclosed. However, in a technique that adds noise to protect individual records, a disclosure risk might be measured by the re-identification risk, a measure used in security of statistical databases.

3. *Information Loss*

   Information loss is usually proportional to the amount of noise added, and the level of security. It is inversely proportional to data quality. One basic requirement of a privacy preserving technique is its ability to maintain high data quality in a released data set. Even the maximum level of privacy protection could turn out to be useless if the data quality is not maintained.

   The information loss is highly dependent on the data mining task for which the data set is intended. For example, in mining association (classification) rules, information loss could be measured by the percentage of rules that have been destroyed/created by the technique, and/or by the reduction/increase in the support and confidence of all the rules; for clustering, information loss can be evaluated by the variance of the distances among the clustered items in the original database and the sanitized database [109].

4. *Cost*

   Cost refers to both the computational cost and the communication cost between the collaborating parties [109]. Computational cost encompasses both preprocessing cost (e.g., initial perturbation of the values) and running cost (e.g., processing overheads). If a data set is distributed then communication cost becomes an important issue in privacy preserving data mining. The higher the cost, the lower the efficiency of the technique.

In Section 3.4 we illustrate these criteria by presenting a comparative study of a few privacy preserving techniques. The techniques have been carefully selected so as to exemplify a broad range of methods. In the next two sections we present a background study on *Data Modification* and *Secure Multi-party Computation* techniques.

3.2 Data Modification

Existing privacy protection methods for centralized statistical databases can be categorized in three main groups, based on the approaches they take, such as query restriction, output perturbation, and data modification [1]. In a way, out of these privacy protection techniques, data modification is the most straightforward one to implement. Before the release of a data set for various data mining tasks and statistical analyses, it modifies the data set so that individual privacy can be protected while the quality of the released data remains high. After this modification of the data set we can use any off the shelf software such as DBMS, and See5 to manage and analyse the data without any restrictions on processing. That is not the case with query restriction and output perturbation. The simplicity of data modification techniques has made it attractive and widely used in the context of statistical database and data mining. Data modification can be done in many ways such as noise addition, data swapping, aggregation, and suppression. We take an opportunity to introduce the basic ideas of these techniques and brief background studies on them.

3.2.1 Noise Addition in Statistical Database

Noise addition techniques were originally used for statistical databases which were supposed to maintain data quality in parallel to the privacy of individuals. Later on noise addition techniques were also found useful in privacy preserving data mining. In this subsection we present background studies on noise addition techniques in statistical databases. We discuss noise addition techniques used for privacy preserving data mining in the next subsection.

In noise addition, generally a random number (noise) is drawn from a probability distribution having zero mean and a small standard deviation. This noise is then added to a numerical attribute in order to mask its original value. Generally noise is added to the confidential attributes, of a microdata file before the data is released, in order to protect the sensitive information of an individual. However, adding noise to both confidential and non-confidential attributes can improve the level of privacy by making re-identification of the records more challenging. The main objective of noise addition is to protect individual privacy by masking the microdata while introducing the least amount of incorrectness in it. The incorrectness in the statistic of a perturbed data set with respect to the statistic of the unperturbed data set is termed as bias. Mulalidhar et al. [73] presented a useful classification of various types of bias as follows.

- Type A Bias - Bias due to the change in variance of an individual attribute.
- Type B Bias - Bias due to the changes in relationship such as covariance, and correlation between confidential attributes.
- Type C Bias - Bias due to the changes in relationship between confidential and non-confidential attributes.
- Type D Bias - Bias due to the change in the underlying distributions of a data set. The underlying distributions of a perturbed data set can be unpredictable if the distributions of the corresponding original data set and/or the distributions of the added noise is not multivariate normal. In such a case responses to queries involving percentiles, sums, conditional means etc. can be biased. Although type A and type D bias occurs for the same types of queries, the reason and extent of bias are different.

Early noise addition techniques were relatively unsophisticated and only protected against bias in estimating the mean of an attribute. Gradually noise addition techniques evolved in order to protect against various biases such as Type A, Type B, Type C and Type D bias.

Early noise addition techniques were applicable only to numerical attributes - since it was not straightforward to add noise to a categorical attribute, which does not have any natural ordering in it. In 1965 Warner proposed a noise addition technique for categorical attributes of domain size two [111]. Warner presented the technique in order to provide privacy for the participants of a survey. A survey participant provides the true value of a categorical attribute with a given probability p which is less than 1. A reasonably accurate estimate of the original proportion of the categorical values can be made from the perturbed values. This technique can also be extended to estimate the distribution of a categorical attribute having domain size bigger than two. However, since noise is added to each attribute separately this technique does not preserve the correlations among the attributes. In 1984 Traub et al. proposed another noise addition technique which provided a balance between the level of security and data quality with the help of Chebyshev's inequality [102]. Nevertheless, it also adds noise to each numerical attribute separately, and consequently the issue of preserving correlations remains unaddressed.

In an attempt to preserve the correlations along with the means and the distributions of numerical attributes, Liew et al. developed another technique in 1985 [68]. They proposed a probability distribution based noise addition technique which involves three steps. In the first step it identifies the density functions of all attributes that need to be perturbed and estimates the parameters of the density functions. In the second step it generates attribute values for each of these attributes from the corresponding density function. Finally, in the third step it maps and replaces the generated values of an attribute for the original values. The first two steps are devoted to the preservation of means and distributions of the attributes while the third step attempts to preserve the correlations among the attributes. However, the method still adds noise to the attributes separately. The preservation of correlations relies on the mapping and replacement step which may not be able to preserve them in a controlled manner. This method suffers from Type A, Type B, Type C, and Type D bias.

Kim, in 1986, proposed a technique that added correlated noise [56]. This technique can be expressed as $Y_{ij} = X_{ij} + e_{ij}$, where Y_{ij}, X_{ij}, and e_{ij} are respectively the perturbed, original and noise value of the *ith* record's *jth* attribute. The covariance matrix of noise is $d * \sum_{XX}$, where $\sum_{XX}$ is the covariance matrix of the attributes of the unperturbed data set and d is a user defined constant. In addition to the correlated noise Kim used a linear transformation to provide better security by minimizing disclosure risk in a released data set. Muralidhar et al. noted that the transformation could also be used to eliminate Type A bias in a released data set [73]. Additionally, since both the data set and the noise are considered to have multivariate normal distribution the perturbed data set is free from Type D bias. However, the perturbed data set still suffers from Type C bias.

In 1994 Tendick and Matloff also used a linear transformation in their proposed noise addition technique, which is also free from Type A, Type B and Type D bias [100]. They pointed out that just the addition of correlated noise is not good enough for the preservation of conditional means. A conditional mean of an attribute is the mean of all values (of the attribute) belonging to a subpopulation that fulfils a particular condition. They also demonstrated the change of distribution of attribute values within a subpopulation due to noise addition. It was suggested that on top of the addition of correlated noise a linear transformation is required to preserve the distributions of attributes and the estimates within a subpopulation.

In 1995 Kim and Winkler proposed a noise addition technique which builds upon the basic concept of the technique proposed by Kim in 1986 [58]. The records having high re-identification risk, even after the addition of correlated noise, are detected. A group of similar records is obtained for each record having high re-identification risk. Attribute values of the risky record are swapped with values of another record belonging to the same group in order to increase the security. Finally, the technique provides a controlled distortion phase where a series of complementary changes are made for any changes of values belonging to a risky record. The controlled distortion phase is meant for preserving the estimates within a subpopulation. They also presented formulae to obtain subpopulation estimates from the data set perturbed by their technique. They assumed that both the data set and the noise had multivariate normal distributions. However, in reality it is not unlikely that a data set would have a general distribution. For data sets having general distribution they suggested a transformation based on the Sullivan and Fuller method proposed in 1989 and 1990. It involves the following steps - transforming general distribution to multivariate normal, masking the transformed data, and transforming back the masked data to the original scale.

In 1999 Muralidhar et al. proposed an advanced noise addition technique called General Additive Data Perturbation (GADP) [73], which was capable of maintaining the marginal distributions of attributes, and relationships among all attributes - provided the data set had a multivariate normal distribution. Although the GADP technique perturbs only the confidential atributes, it takes both confidential and non-confidential attributes into account. Therefore, it maintains the relationships among confidential and non-confidential attributes. The GADP technique allows a database administrator to precisely specify the desired relationship among attributes after perturbation. It also allows the database administrator to specify a desired level of security in the perturbed data set. GADP technique is free from bias of Type A, B, C and D when it is applied on a data set having multivariate normal distribution. However, when applied on a data set having non-normal distribution, GADP technique can not preserve the marginal distributions of confidential attributes and/or the relationships between the attributes [92].

In 2002 Sarathy et al. proposed a copula based GADP technique called C-GADP, which can preserve the marginal distributions, and monotonic relationships among all attributes, even if the data set does not have a multivariate normal distribution [92]. A copula is a function which takes as input the marginal distributions of individual attributes and outputs the multivariate distribution of the data set. The basic idea of C-GADP is similar to what was suggested in [58]. It involves three basic steps; transformation of existing distributions to normal distributions, masking the transformed data by GADP technique, and finally transformation of the masked data back to its original form. Sarathy et. al. reported that the C-GADP technique may not preserve non-monotonic relationships among attributes. If two attributes **X** and **Y** have a $\bigcup$-shaped or $\bigcap$-shaped relation between them, C-GADP may not preserve it although **X** is purely a function of **Y**. Sarathy et al. also reported that they used a normal copula which may not preserve “tail dependence” i.e. dependence in extreme values of a bivariate distribution.

GADP lacks the ability to maintain statistical estimates of a data set having small number of records, even if the data set has a multivariate normal distribution. Therefore, Muralidhar et al. proposed an enhancement to GADP, Enhanced General Additive Data Perturbation (EGADP), which performs well on both small and large data sets [74]. For more information on inference problem in statistical database we refer the reader to surveys [1, 33].

3.2.2 Noise Addition in Data Mining

The noise addition techniques discussed in the previous section were designed for statistical databases and did not take into account requirements specific to data mining applications. In 2002, Wilson and Rosen [116] investigated the prediction accuracies of classifiers built from data sets perturbed by existing statistical noise addition techniques such as SADP (Simple Additive Data Perturbation), and GADP. They found that the classifiers obtained from data sets perturbed by noise addition techniques including GADP, suffer from a lack of prediction accuracy on a testing data set. This suggests the existence of another type of bias, Data Mining bias, which is related to the change of patterns of a data set. Patterns of a data set include clusters, classification and association rule sets, and subpopulation correlations.

In 2000 Agrawal and Srikant proposed a noise addition technique which added random noise to attribute values in such a way that the distributions of data values belonging to original and perturbed data set were very different [3]. In this technique it is no longer possible to precisely estimate the original values of individual records. However, a reconstruction procedure is used to regenerate the distribution of the original values. Based on the reconstructed distributions a decision tree is built. It has been shown that the decision tree exhibits a good prediction accuracy, even for higher levels of noise. An advantage of this technique is that it is also applicable to distributed data sets, as every party is free to add random noise to its data set before sharing it with other parties. The technique can also be applied in the scenario that has been considered in [111, 26, 97]. In this scenario every survey participant can add random noise to his/her data before submitting the randomized response to the central server. According to the technique the central server can still reconstruct the distribution and thereby produce a reasonably accurate classifier. Although each individual response is randomized, if the number of respondents is large the distribution of the original data can be estimated with sufficient accuracy.

Du and Zhan presented a decision tree building algorithm which is used on a perturbed data set, at a central server, collected through randomized responses of the clients/survey-participants [26]. They also proposed a randomized response technique that can be used to perturb multiple attributes. The proposed tree building algorithm is a modification of ID3. ID3 algorithm discovers the best split point of the best splitting attribute at each node of a decision tree. For that purpose ID3 calculates the entropy and the information gain of each split point from the data set. Since the proposed algorithm is applied on a perturbed data set it employs a probabilistic estimation of the original proportions of records . Thereby it calculates the original information gain of a split point using the perturbed data set. Finally it builds a decision tree having high prediction accuracy on an unperturbed testing data set. Nevertheless, the technique restricts all attributes to have only binary data unlike the distribution reconstruction based technique of Agrawal et. al [3]. Both of the techniques focus on building decision trees and thereby producing a predictor.

Unfortunately any reconstruction of an original distribution, such as the one proposed by Agrawal and Srikant [3], suffers from information loss. In order to minimize this information loss Agrawal and Aggarwal developed a novel reconstruction algorithm called

Expectation Maximization (EM) algorithm in 2001 [2]. This algorithm converges to the maximum likelihood estimate of the original distribution. The EM algorithm works better for data set having large number of records. Agrawal and Aggarwal proposed a privacy and information loss quantifying method which considered the reconstructed distribution along with the perturbed data as accessible information of an intruder. However, the information loss quantifying method of Agrawal and Srikant [3] considered only the perturbed data as accessible. Agrawal and Aggarwal showed that under the former consideration an intruder can guess a data value with a higher level of accuracy.

In 2003 Evfimievski et al. expressed their hesitation in the strength of these distribution reconstruction based techniques in privacy preservation [31]. They argue that based on the distribution of noise one may be able to learn with high confidence that some of the records satisfy a particular property. For example, when someone knows that the perturbed value of an attribute "age" is 120 and the noise is drawn from a uniform distribution ranging from -50 to +50, then s/he can infer with 100% accuracy that the actual age is at least 70. This type of inference is known as interval based inference.

An interval based inference occurs when a reasonably small interval, into which the sensitive value must fall, can be estimated. The effectiveness of noise addition techniques was questioned, also by Li et al., in preventing interval based inference in the context of sum queries [66]. They proposed a noise addition technique based on a new type of distribution, called ϵ-Gaussian distribution, instead of traditional Gaussian distribution. Evfimievski et al. proposed a formulation of privacy breach based on the difference in knowledge of an intruder before and after the release of a perturbed value [31]. They also proposed a methodology called "amplification" to limit the breach of privacy, unless any knowledge of the distribution of original data set is released. "Amplification" is catered for association rule mining problem, making use of a modification of the algorithm proposed by Evfimievski [32].

In 2004 Zhu and Liu [121] proposed a general framework for randomization using a well studied statistical model called mixture models. According to this scheme perturbed data are generated from a distribution (not from the addition of noise to original data) that depends on factors including the original data. Their randomization framework supports privacy preserving density estimation. It estimates the original data distribution from the perturbed data. Zhu and Liu showed that the noise addition schemes used in some other studies [3, 2, 31] were special cases of their framework. A significant drawback of these techniques is that a data miner can only learn the original distributions of attribute values from a perturbed data set. S/he can not expect to extract other original information such as classification rules and clusters of records. Some data swapping and noise addition techniques allow a data miner to perform better data investigation.

Zhu and Liu argue that their framework is flexible in the sense that data and noise can be continuous or discrete and they can be of different types, i.e. one discrete and another continuous. Moreover, the distribution from which the perturbed data are drawn can be general or any particular type such as normal distribution.

Kargupta et al. questioned the usefulness of additive noise perturbation techniques in preserving privacy [54, 53, 20, 55, 72]. They proposed a spectral filtering technique which makes use of the theory of random matrices to produce a close estimate of an original data set from the perturbed (released) version of the data set. They assume that the distribution of noise and the perturbed data set are known to the user. From the distribution of noise the theoretical bounds of eigenvalues, λ_{min} and λ_{max} for the noise matrix are estimated - using fundamental properties of random matrices. The noise matrix is the matrix, elements of which are the random noise values added to the original data set. The filtering technique then calculates the eigenvalues and the eigenvectors of the covariance matrix of the released data set.

Out of these eigenvalues those which fall within the range of λ_{min} and λ_{max} are the noisy eigenvalues, whereas the remaining are the eigenvalues related to the actual data set. Based on these two groups of eigenvalues and corresponding eigenvectors, the covariance matrix of the released data set is decomposed into two matrices. One matrix is related to the random noise part and the other matrix is related to the actual data part. Finally a close estimate of the original data set is made - by using the released data set along with the eigenvalues, the eigenvectors and the covariance matrix all related to the actual data part.

We argue that if the privacy is defined as the uncertainty in estimating a sensitive attribute value belonging to an individual, then noise addition techniques can still be considered safe. This is because of the fact that the filtering technique can not generate an estimate of the original data set with 100% accuracy, resulting in an uncertainty in estimating original values. Besides, the filtering technique is based on some assumptions including the one that considers the added noise and the unperturbed data set are uncorrelated. This is the case for some noise addition techniques such as the one proposed by Agrawal [3], but in many techniques [73, 92, 58] they are correlated.

Filtered data set is obtained from a perturbed data set using a filtering technique. Although it is unlikely to be able to reproduce an original data set using a filtering technique, generally the data quality of a filtered data set could be better than the data quality of the perturbed data set - depending on the amount and type of noise added during the perturbation. For example, mean of an attribute in a filtered data set (compared to the mean of the attribute in the perturbed data set) can be close to the mean of the attribute in the original data set. The lower the amount of noise the better the quality of a filtered data set should be. We believe that a filtering technique can better be used on a released data set by an honest data user who is mainly interested in the extraction of general patterns and statistics.

Kargupta et al. conjectured that multiplicative noise could be a better privacy preserving data modification technique. In 2003 Kim and Winkler investigated the data quality and the level of privacy in a data set perturbed by multiplicative noise [59]. They considered two multiplicative noise perturbation techniques. The first technique involves the generation of random numbers having mean 1 and a small variance, and multiplication of the original data by these generated numbers. The second technique embraces the following steps - logarithmic transformation of original data, computation of covariance matrix of the transformed data, generation of random numbers having mean 0 and variance/covariance equals to c (a user defined constant) times the variance/covariance of the transformed data, addition of the generated noise to the transformed data, and performing an antilog of the noise added data. Kim and Winkler apply the second technique with two different c values, c=0.01 and c=0.10. They mask a data set by all these techniques. Means and variances of masked data sets are compared with the means and variances of the unmasked data set. These comparisons are made in order to evaluate the usefulness of the techniques in maintaining the data quality. However, neither of these techniques is found to be clearly superior to the other.

We note that if the results presented by Kim [59] are studied and compared with the some other results proposed by Muralidhar et al. [74, 92, 73], then they clearly explain the relative inferiority of multiplicative noise techniques in relation to additive noise techniques in maintaining data quality. Multiplicative noise techniques can provide a good level of privacy as conjectured by Kargupta et al. [55], but the additive noise perturbation techniques are more effective in maintaining data quality which is a primary requirement of any data sets. Therefore, noise addition techniques are widely used in almost all areas of privacy preserving data mining including classification, clustering, and association rule mining.

The concept of association rule mining has been discussed in Chapter 2 in details. Many privacy preserving association rule mining techniques have been proposed recently [82, 110, 89, 83, 80, 81, 120, 96, 94]. In 2002 Evfimievski et al. proposed a novel noise addition technique for privacy preserving association rule mining [32]. In addition to the perturbation of the items, the technique also inserts many false items into a transaction in order to reduce the disclosure risk of an item set belonging to the transaction. Additionally, a method was also proposed for estimating an item set's support in the original data set while only the perturbed data set is available. Evfimievski et al. also gave a formal definition of privacy breach. If the appearance of an item set A in a perturbed transaction makes the probability of the appearance of an item $a \in A$ in the corresponding unperturbed transaction higher than a user defined threshold θ, then disclosure of A is a privacy breach of level θ. Evfimievski showed that the noise addition techniques proposed by R. Agrawal et al. and D. Agrawal et al. [3, 2] may not be good enough to protect privacy in the context of association rule mining, if the number of possible items is very big. It is possible that an original item set can be left unchanged in several transactions of a perturbed data set having a huge number of transactions, even if the original items are perturbed with high probability. On the contrary it is almost impossible that an item set, created artificially as a result of randomization, can appear even in few transactions - if the number of possible items (domain of items) is very big. Therefore, if an item set appears in a notable number of transactions then an intruder can presume its existence in the corresponding unperturbed transactions with high certainty. Thus, the privacy of some transactions could be infringed.

In 2002 Rizvi and Haritsa also used the same approach of insertion of false items in order to preserve privacy in a market basket database [89]. In a market basket database rows are the purchase records of customers whereas, columns are the items sold by a supermarket. Each row is represented by a fixed-length series of 1s and 0s, where 1s represent purchase of corresponding items and 0s represent no purchase. Rizvi and Haritsa proposed a perturbation technique which left the purchase status of an item unchanged with a user defined probability p, and flipped it with a probability $1 - p$. Flipping a 0 to 1 means insertion of a new item whereas, flipping a 1 to 0 means deletion of an existing item. They pointed out that the disclosure of a 1 is more serious privacy breach than the disclosure of a 0.

The techniques discussed so far try to protect the disclosure of any item set, belonging to a transaction, with high certainty while they aim to mine accurate association rules. However, in some scenario the perturbation techniques aim to protect some sensitive association rules from being disclosed. An example of such scenario is multi-party cooperative association rule mining, where few organizations share their transactional databases for rule mining. This cooperation usually benefits each organization in mining more fruitful association rules. However, there could be some sensitive rules hidden within a database belonging to an organization, which may not want to share the rules with others. Therefore, disclosure of such rules may be considered as a breach of privacy.

In order to hide the sensitive rules of a data set noise is added (i.e. few items are changed or deleted) in a way so that the supports of the sensitive rules become lower than the minimum support level. This idea was introduced by Atallah et al. in 1999 [8], and has been used in several techniques [83, 82, 81]. In 2003 Oliveira and Zaïane proposed a perturbation technique that took the similar approach to hide sensitive rules [83]. The proposed algorithm needs only one scan over the transactional data set regardless of the size of the data set, and the number of sensitive rules that need to be protected. For a given sensitive rule the algorithm chooses an item for perturbation. The chosen item is perturbed in the required number of transactions among the list of transactions that support the rule. Shorter sized transactions have the lesser combinations of rules. Therefore, shorter transactions are chosen for the perturbation to minimize the impact of noise addition. Besides, the algorithm checks if there is an item common between two or more sensitive rules. If there is such a common item then the algorithm chooses to perturb that item in order to hide all rules having the item in common. This way the number of modifications are minimized in order to maintain both high data quality, and privacy of sensitive rules.

Natwichai et al. proposed an algorithm for hiding classification rules [76]. All classification rules discovered from an original data set are presented to the data owner. He/she detects the sensitive rules. Based on the remaining non-sensitive rules and other extracted properties of the data set a decision tree is built, which is eventually used to generate a perturbed data set.

Saygin et al. perturbed an item by replacing it with an unknown value i.e. a "?" symbol instead of modifying/deleting the item - for preserving privacy of sensitive rules [94, 93]. They argued that in many situations it could be desirable to replace an item by an unknown value instead of a false value. Replacing the original item by a false one can guide to obtain misleading rule which can be dangerous in some cases. For example, if such misleading rules are obtained from a medical data set used for diagnosis purpose, it can create a serious harm. Saygin et al. proposed algorithms to preserve privacy of sensitive rules by reducing their support, and by reducing their confidence in the perturbed data set [94].

Agrawal and Haritsa [4] proposed a framework for designing perturbation techniques. They showed that many prior techniques differ only in the elements of a matrix used in their framework and that new ones can be constructed by choosing appropriate matrix elements. They proposed a technique that provides highly accurate mining results for identifying frequent item sets. Agrawal et al. [5] showed that mining of association rules can take significantly longer time on a perturbed data set than on an original data set. They proposed an algorithm for achieving high degree of privacy and accuracy requiring relatively shorter time for mining.

3.2.3 Data Swapping

Data swapping techniques were first devised by Dalenius and Reiss in 1982, for categorical values modification in the context of secure statistical databases [18]. The main appeal of the method was it keeps all original values in the data set, while at the same time makes the record re-identification very difficult. The method actually replaces the original data set by another one, where some original values belonging to a sensitive attribute are exchanged between them. This swapping can be done in a way so that the t-order statistics of the original data set are preserved. A t-order statistic is a statistic that can be generated from exactly t attributes. In [86] a new concept called "approximate data swap" was introduced for practical data swapping. It computes the t-order frequency table from the original data set, and finds a new data set with approximately the same t-order frequency. The elements of the new data set are generated one at a time from a probability distribution constructed through the frequency table. The frequency of already created elements and a possible new element is used in the construction of the probability distribution. An introduction to the existing data swapping techniques can be found in [75, 35].

Inspired by existing data swapping techniques used for statistical databases a new data swapping technique has been introduced for privacy preserving data mining, where the requirement of preserving t-order statistics has been relaxed [28]. The technique emphasizes the pattern preservation instead of obtaining unbiased statistical parameters. It preserves the most classification rules, even if they are obtained by different classification algorithms. The noise is added to the class, i.e. the target attribute of a classifier, instead of all other attributes in the data set. As the class is typically a categorical attribute containing just two different values, the noise is added by changing the class in a small number of records. This is achieved by randomly shuffling the class attribute values belonging to heterogeneous leaves of a decision tree. If a leaf corresponds to a group of records having different class attribute values, then the leaf is known to be a heterogeneous leaf.

3.2.4 Aggregation

Aggregation is also known as generalization or global recoding. It is used for protecting an individual privacy in a released data set by perturbing the original data set prior to its release. Aggregation replaces k number of records of a data set by a representative record. The value of an attribute in such a representative record is generally derived by taking the average of all values, for the attribute, belonging to the records that are replaced. Due to the replacement of k number of original records by a representative record aggregation results in some information loss. The information loss can be minimised by clustering the original records into mutually exclusive groups of k records prior to aggregation. However, a lower information loss results in a higher disclosure risk since an intruder can make a better estimate of an original value from the attribute value of the released record. An adjustment of the cluster size i.e. the number of records in each cluster can produce an appropriate balance of information loss and disclosure risk [67]. Another method of aggregation or generalisation is transformation of attribute values. For example, an exact date of birth can be replaced by the year of birth, an exact salary can be replaced rounded in thousands. Such a generalisation makes an attribute values less informative. Therefore, an use of excessive extent of generalisation can make the released data useless. For example, if an exact date of birth is replaced by the century of birth then the released data can become useless to data miners [49].

3.2.5 Suppression

In suppression technique sensitive data values are deleted or suppressed prior to the release of a microdata. Suppression is used to protect an individual privacy from intruders' attempts to accurately predict a suppressed value. An intruder can take various approaches to predict a sensitive value. For example, a classifier, built on a released data set, can be used in an attempt to predict a suppressed attribute value [48]. Therefore, sufficient number of attribute values should be suppressed in order to protect privacy. However, suppression of attribute values results in information loss. An important issue in suppression is to minimize the information loss by minimizing the number of values suppressed.

For some applications, such as medical, suppression is preferred over noise addition in order to reduce the chance of having misleading patterns in the perturbed data set. Suppression has also been used for association and classification rule confusion [89, 94].

3.3 Secure Multi-Party Computation

In many situations mining the joint data belonging to several parties is crucial. At the same time keeping the data owned by individual parties private is compulsory due to various reasons such as laws, mutual untrusts and interests of the parties. Examples of such situations include cooperation among different health service providers, manufacturers, and government bodies such as a taxation office and a social security department. Health service providers such as hospitals can be interested to mine their joint data for better diagnosis and treatment. However, they may not be able to disclose their data due to some legal obligation for preserving privacy of the patients. Similarly, even competing manufacturers may want to mine their joint data to discover interesting relations among their manufactured items. They may be interested to see which of their items are used together in a final product. Nevertheless, they can be reluctant to disclose their data in order to protect their private business information. A taxation office and a social security office may need to mine their joint data set for discovering interesting patterns involving both data sets - without disclosing them.

Data modification for privacy preserving data mining takes the approach to modify data before releasing it. In this approach once the data set is released the user has complete freedom to use the high quality data for any kind of analysis. Multiple parties having different portions of a data set can also perform data modifications on their corresponding data sets for getting a more useful combined data set [32, 89]. However, data modification techniques do not protect every single information other than the final result of a computation. However, such protection could be a basic requirement for a multi-party cooperation in some scenarios. In the absence of a reliable privacy protection technique to fulfill this requirement, parties may not have any option to cooperate. As a result they are forced to deprive themselves from the benefits of data mining. Therefore, techniques based on a more conservative cryptographic approach are often used in this scenario. These techniques are commonly known as Secure Multi-party Computation (SMC) techniques.

Any computation performed by two or more mutually untrusted parties can be termed as Secure Multi-Party Computation (SMC), where parties are interested to compute a result from the union of data owned by each individual party [23]. However, none of the parties wants to reveal its data to any other party. The concept of SMC was originally introduced by Yao in 1982 [119]. It has been extensively studied since then. Ideally, SMC is supposed to reveal to a party just the result of the computation and the data owned by the party. However, in practical applications this requirement is sometimes relaxed in order to have a better performing algorithm, considering a disclosure of some insensitive information as acceptable [17]. SMC has an application in various areas including privacy preserving data mining, security of statistical databases, and private information retrieval. A number of SMC algorithms for various data mining tasks has been presented in last few years [41, 104, 51, 108, 106, 118, 25, 24, 71, 70, 105, 91, 107, 84]. Most of these algorithms make use of some primitive computations such as secure sum, secure set union, secure size of set intersection and secure scalar product.

Secure sum adds values stored in different parties without letting them know any new information apart from the sum of the values. If there are s number of parties, where the party i has a value x_i - then secure sum discloses the value of $x = \sum_{i=1}^{s} x_i$ to all parties. Secure set union computes the union of items belonging to a group of parties. Similarly secure size of set intersection discloses $|S_1 \bigcap S_2 ...... \bigcap S_n|$, where S_i is the set of items belonging to the party P_i. Secure scalar product calculates $\sum_{i=1}^{n} x_i \times y_i$, where party P_1 has the vector $\overrightarrow{X} = \langle x_1, x_2, ...x_n \rangle$ and party P_2 has the vector $\overrightarrow{Y} = \langle y_1, y_2, ...y_n \rangle$ [17].

Various data mining tasks can be reduced to these primitive computations. For example, privacy preserving association rule mining can be reduced to a secure scalar product [104, 17]. An essential step for association rule mining is to compute the support count for an item set I_s. If we find the support count is greater than a given threshold, then I_s is known as a frequent item set which is used in mining association rules. Now, let us consider that a market basket data set is vertically partitioned into two data sets belonging to two different parties. Items of each transaction are split between the parties. The support count of I_s is exactly the same as the scalar product of the two vectors $\overrightarrow{X} = \langle x_1, x_2, ...x_n \rangle$, and $\overrightarrow{Y} = \langle y_1, y_2, ...y_n \rangle$ belonging to the parties, where n is the number of transactions. This is illustrated as follows.

Suppose party 1 has an item set $\{a_1, ...a_k\}$ of size k and party 2 has another item set $\{b_1, ...b_l\}$ of size l. Let, the subset of the item set I_s belonging to party 1 be $\{a_1, a_2, a_3\}$, and the subset belonging to party 2 be $\{b_1, b_2\}$. Suppose z_{ij} is the value of the jth item of the ith transaction. The meaning of $z_{ij} = 1$ is the presence of the jth item in the ith transaction, and $z_{ij} = 0$ means the absence of the item in the transaction. An element of the vector $\overrightarrow{X} = \langle x_1, x_2, ...x_n \rangle$, x_i is computed as $\prod_{j=1}^{3} a_{ij}$. Similarly an element of the vector $\overrightarrow{Y} = \langle y_1, y_2, ...y_n \rangle$, y_i is computed as $\prod_{j=1}^{2} b_{ij}$. The scalar product of the vectors $\overrightarrow{X}$ and $\overrightarrow{Y}$ can be computed as $\overrightarrow{X}.\overrightarrow{Y} = \sum_{i=1}^{n} x_i * y_i$. The result of the computation is nothing but the support count of the item set I_s in the combined data set. Therefore, a privacy preserving association rule mining can be reduced to a secure scalar product. Similarly many other privacy preserving data mining tasks can be reduced to primitive computations of the SMC problem. For horizontally partitioned data the computation of frequent itemsets reduces to the computation of secure set union [51].

Moreover, secure sum is used in privacy preserving data mining such as privacy preserving association rule mining in horizontally partitioned data sets [17]. Secure sum is often used to demonstrate the concept of SMC. We illustrate the basic idea of secure sum as follows. Suppose there are s number of parties, where $s \geq 3$, and the party i has the value x_i. Secure sum computes the value of $x = \sum_{i=1}^{s} x_i$, without letting any party know any new information apart from the result of the computation. The sites are numbered uniquely from 1 to s. The site 1 generates a random number R uniformly chosen from the range $[0..n]$, where the sum x is known to have a range $[0..n]$. Site 1 then computes $R_1 = (R + x_1)$ mod n, and sends R_1 to site 2. Since R is chosen uniformly from the range $[0..n]$, R_1 is also uniformly distributed over the same range. Thus, site 2 does not learn anything about the value x_1. Site 2 similarly computes $R_2 = (R_1 + x_2)$ mod n, and sends R_2 to site 3. Finally site s receives R_{s-1} from site $s - 1$. It computes $R_s = (R_{s-1} + x_s)$ mod n, and sends R_s to site 1. Site 1 then calculates $x = (R_s - R)$ mod n, and sends x to all parties. Note that each party is assumed to have used its correct value x_i. Thus, secure sum ensures that all parties can learn the final result of the computation without learning anything else new - unless there is a collusion.

Site $(k-1)$ and site $(k+1)$ are two neighbors of site k. If these two neighbors collude then it is possible for them to disclose the value of site k. Site $k-1$ sends R_{k-1} to site k, and site k sends R_k to site $k+1$. From this information the colluding neighbors can precisely learn the value of site k, $x_k = (R_k - R_{k-1})$. This kind of security threat can be prevented by extending the protocol, provided majority of the parties/sites are honest. According to the extended protocol each site divides its value into m shares. The sum of each share is computed separately. Finally the ultimate result can be computed by adding the sums of all shares. The ordering of sites are changed for each share, so that none of the sites has the same neighbor twice. By increasing the number of shares we can increase the required number of dishonest parties to subvert the protocol. Collusion is a serious threat to secure multi-party computation, especially when a big number parties participate in the collusion.

An association rule mining algorithm is proposed in [41] which is secured against malicious participants. It introduces the concept of k-privacy, which restricts the disclosure of any information involving less than k participants. However, the algorithm suffers from the drawback of not considering collusion.

Many SMC techniques have been presented for performing various data mining tasks such as classification, clustering, and association rule mining [70, 50, 104]. Clifton et al. note that different data mining tasks often perform similar elementary computations at various stages [17]. They argue that presenting a separate technique to each specific data mining task may not be a feasible solution for the industries in real world. Therefore, they initiate to build a toolkit having a number of components that can be assembled in various ways for developing more complex, application specific, privacy preserving techniques. Such a sophisticated toolkit can make various distributed privacy preserving data mining tasks industry feasible. This idea is also supported in [51]. We briefly describe some privacy preserving data mining techniques in the following paragraphs.

Secure multi-party computation for mining association rules has been studied in [41, 104, 51, 108]. The task here is to develop a SMC for computing the global support count, of an itemset, which can be used to find the global frequent itemsets (GFIs). GFIs are the itemsets having global support count greater than a user defined threshold. The technique proposed in [108] relies on the fact that a GFI has to be a frequent itemset in at least one of the partitions of a horizontally partitioned data set. In this technique, all maximal frequent itemsets (MFIs) of each partitions are locally computed by the parties, where an MFI is a frequent itemset which is not a subset of another frequent itemset. The union of all these local MFIs is then computed by a trusted third party, which takes encoded local MFIs from the parties as inputs. The support counts of all possible subsets of each of the MFIs belonging to the union are computed by the parties locally. Finally, the global support count for each itemset is computed by summing up all the local support counts for the itemset, using the secure sum computation. GFIs can be used for various purposes such as the discovery of association rules, strong rules, correlations, and sequential rules.

A secure multi-party computation function for naive Bayesian classifier on horizontally partitioned data that relies on the secure sum was proposed by Kantarcoglu and Vaidya [51]. They also proposed an extension that takes a logarithm based approach to enhance the security. Building a decision tree on horizontally partitioned data, based on oblivious transfer protocol was proposed by Lindell and Pinkas [70]. The protocol uses the well known ID3 algorithm for building decision trees. Each party performs most of the computations independently, on the database that it owns. This increases the efficiency of the protocol. Finally the results obtained from these independent computations are combined using a cryptographic protocol.

Secure functions for vertically partitioned data relying on secure scalar product were proposed in [106, 118]. The protocol proposed by Vaidya and Clifton builds a model where each party has a random share of the model [106]. The parties collaborate to classify an instance. Only the class of each instance is disclosed as a result of the use of the protocol. The model generated from the combined data is not disclosed. Such a discloser is often unwanted due to legal and/or commercial issues. However, the classifier can be reverse-engineered from the knowledge of class values belonging to a sufficient number of instances. A solution to building a decision tree on vertically partitioned data was presented by Du and Zhan, based on a secure scalar product [25]. In order to increase the performance, they used a semi-trusted commodity server that belongs to a third party.

A method for secure multi-party computation of k-means clustering on vertically partitioned data was presented by Vaidya and Clifton [105]. The method makes use of some secure primitive computations such as secure sum, and secure comparison. Regression on vertical data was considered in [91, 24], while secure computing of outliers for horizontally and vertically partitioned data was studied in [107].

3.4 Comparative Study

Generally Secure Multi-party Computation techniques tend to incur a significantly higher running and communication cost, but also to provide much higher level of security. It does not disclose anything other than the final result such as the classification rules, clusters, and association rules. Therefore, it is suitable in a particular scenario where multiple parties agree to cooperate for just the final result extraction from their combined data set. However, in a scenario where a data set is supposed to be released to facilitate various research and extract general knowledge, Data Modification is the obvious choice. The definition of privacy breach is also relaxed in this scenario. Data modification usually incur less computational cost, and less information loss as well. In Table 3.1, we compare few techniques against the evaluation criteria presented in Section 3.1.

3.5 Conclusion

In this chapter, we introduced privacy preserving data mining and briefly presented two classes of techniques: Data Modification and Secure Multi-party Computation. These techniques are used in a wide range of scenarios. In the next four chapters we present a novel privacy preserving technique which uses a Data Modification approach. The technique considers the class attribute of a data set as sensitive, and can be applied on data sets having both categorical and numerical attributes or any of them. It preserves the patterns of classification rules along with the statistical parameters in the released data set. In Chapter 4 we present techniques for noise addition to a class attribute.

Name	Versatility				Discl. Risk	Info. Loss	Cost
	Private: Data /Rules	Dataset: Central /Distributed (Horizontal /Vertical)	Attributes: Categorical /Numerical /Boolean	Data Mining Task			
Outlier Detection [106]	Data (both)	Distributed	Both	Outliers	Very Low	None	High
Association Rules [104]	Data	Distributed (vertical)	Boolean	Assoc. Rules	Very Low	None	Mod.
Randomized Noise [3]	Data	Both	Numerical	Class.	Low	Mod.	Low
Correlated Noise [73]	Data	Centralized	Numerical	Clust., Class.	Mod.	Low	Low
Decision Tree Noise [28]	Data	Centralized	Both	Class.	Mod.	Low	Low
Randomized Response [26]	Data	Both	Binary	Class.	Low	Mod.	Low
Randomized Response [121]	Data	Both	Numerical	Density Esti.	Low	Mod.	Low
Secure Toolkit [17]	Data	Distributed (Both)	Numerical	Assoc. rules, & Clust.	Low	None	Mod.
SMC [104]	Data	Distributed (Vertical)	Numerical	Assoc. rules	Very Low	None	Mod.
SMC [51]	Data	Distributed (Horizontal)	Numerical	Class	Very Low	None	Mod.
Association Rules [83]	Rules	Centralized	Boolean	Assoc. Rules	Low	Mod.	Low

Table 3.1: Privacy Preserving Techniques - Comparative Study

Chapter 4

Class Attribute Perturbation Technique

4.1 The Essence

Considerations

We consider a data set as a two dimensional table, where the rows (records) correspond to individuals (cases) and columns (attributes) correspond to the properties that describe an individual. Out of these attributes one is the class attribute, which represents the class or category of a record. Typically, it is a categorical attribute with a small domain size. For example, in a patient data set "diagnosis" can be a class attribute, having the domain {HIV positive, HIV negative} and the non-class attributes can be various properties of the patients. An example of such a data set is the *Wisconsin Breast Cancer (WBC)* data set available from the UCI Machine Learning Repository [77].

Such a data set often needs to be released to different parties for various purposes such as research and treatment. These parties need to have access to the data set for performing various data mining and statistical analyses. Since one of the most commonly used data mining technique is classification, we consider that the main purpose of such a release is to allow the users to perform classification by a decision tree.

We primarily consider that a class attribute value, such as "HIV Positive", belonging to an individual is confidential. Therefore, the disclosure (with 100% certainty) of the class attribute value belonging to an individual is considered as a breach of privacy. Such a disclosure is possible through record re-identification by a malicious data miner having some supplementary knowledge about an individual. An individual record can uniquely or almost uniquely be identified by some attributes such as Social Security Number, Driver License Number, and Name. We assume that these identifying attributes are removed from the data set before it is released. However, such exclusions may not be sufficient to protect the privacy of an individual when a combination of other attributes can uniquely identify the record. A malicious data miner (intruder) can have some supplementary knowledge such as ethnic background, religion, and marital status of an individual.

Approach Taken Towards Privacy Protection

Since the class attribute value belonging to an individual is confidential, it should not be learnt by users of the data set. Although data miners need to have unrestricted access to the data set, we argue that they do not need an access to data that is 100% accurate (in fact, that is never the case due to the existence of natural noise in a data set). Therefore, we propose noise addition to a data set in order to protect individual privacy.

We add noise in two steps; in the first step, following [28, 12], we add noise to sensitive class attribute values. This prevents a malicious user from learning the class value belonging to an individual with 100% certainty, even if he/she can re-identify the record from a released data set. Additionally, it may not be straightforward to re-identify a record if there exist other records having the same or similar value in each of the attributes known to the intruder.

However, typically a re-identification is still possible, especially for a data set having a big number of attributes. Therefore, to prevent a re-identification with a high certainty, we in the next step add noise to all non-class attributes, both categorical and numerical. This provides a higher level of privacy, since an intruder is challenged in both re-identifying a record and learning the sensitive class value even if the record is re-identified. Our goal is to add noise in such a way so that the data quality, including patterns of a data set, are preserved. We consider that a perturbed data set is released with an unrestricted access to it. Moreover, the noise addition technique and all parameters for noise addition, such as mean, standard deviation and distribution of numerical noise, are also released.

By a breach of privacy we mean disclosure of a confidential class value belonging to an individual with 100% or very high (close to 100%) certainty. Therefore, if there is a sufficient level of uncertainty in re-identification and thereby learning the class value, then we considered that privacy is being protected. Let us illustrate the concept on an example as follows. Suppose, Alice becomes suspicious about whether or not Bob earns more than 80K. Assume that Alice knows that the record belonging to Bob exists in a released data set which has been modified by the above mentioned noise addition approach. The data set has a class attribute "Income" and many non-class attributes. Suppose that she has unrestricted access to the released data set and she attempts to learn the sensitive class attribute value of the record belonging to Bob through record re-identification. Even if she learns that Bob earns more than 80K, still she can not be 100% certain since she knows that the data set has been modified. There is a chance that she has made a wrong re-identification and/or has obtained a wrong value of the class attribute. She can not come to any definite conclusion based on the information obtained from the released data set. Therefore, we consider that Bob's privacy is protected in the released data set.

We evaluate our methods by using two indicators, data quality and the level of security in a perturbed data set. The data quality of a perturbed data set is measured in two ways. As the first measure we use the similarity between the decision trees produced from the original data set and the perturbed data set. We evaluate the similarity of trees by comparing the logic rules associated with them. As the second measure we use the prediction accuracy of a classifier obtained from a perturbed data set. We measure security of our methods by the uncertainty that users have in estimating the confidential class value.

In this chapter we discuss noise addition to a class attribute. In Chapter 5 and Chapter 6 we discuss noise addition to numerical, and categorical non-class attributes respectively.

4.2 Noise Addition to Class Attribute

Notation

We recall that in a decision tree each internal node represents a test on an attribute, each branch protruding from the node denotes an outcome of the test and each leaf node represents a class or class distribution [47]. Figure 4.1 is an example of a decision tree. The tree has four leaves. Leaf 1, Leaf 2 and Leaf 4 are heterogenous leaves. Records belonging to such a leaf have different class values. However, the majority of the records belonging to a heterogeneous leaf have the same class value. We call these records "majority records" and the corresponding class value "majority class". Similarly, the remaining records are called "minority records" and their class values are called "minority classes". However, all records belonging to a homogeneous leaf have the same class value.

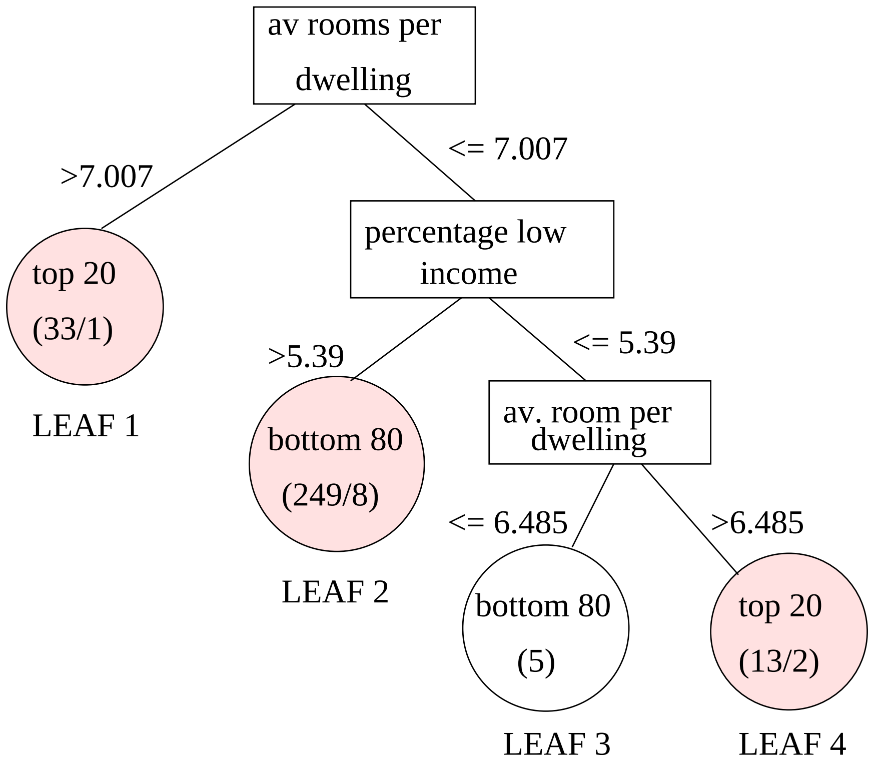

Figure 4.1: An example of a decision tree classifier.

For example, there are altogether thirty three records, belonging to the heterogeneous Leaf 1, out of which thirty two records have the class value "top 20%", while one record has the value "bottom 80%". The class values "top 20%" and "bottom 80%" are termed as majority class and minority class of the leaf, respectively. On the other hand, all records belonging to the homogeneous Leaf 3 have the same class value.

We use the following notation.

H - the number of heterogeneous leaves

m_k - the number of majority records in the kth heterogeneous leaf, $1 \leq k \leq H$

n_k - the number of minority records in the kth heterogeneous leaf, $1 \leq k \leq H$

$E(N)$ - the expected number of changed class values

We first build a decision tree from the unperturbed data set. Following [28, 12], we perturb the class values of the records belonging to the heterogeneous leaves of the decision tree. The records from homogeneous leaves of the original tree remain unchanged. Finally, the perturbed data set is released to data miners.

We argue that if a record belongs to a homogeneous leaf then the value of the class attribute is consistent with a strong pattern identified by the decision tree, and it is very difficult to hide it. For example, if it is a common knowledge that all citizens must retire by the age of 60 and after that they receive a fixed amount of living assistance from the government, then there is nothing to hide about the monthly salary/income of a person who is over 60 years of age. In this case the pattern is very strong and it is likely to be commonly known. Moreover, there will be no difference between the confidentiality of the records from the training set and any other records (not in the training set) due to the very fact that the age over 60 determines the salary.

Three Techniques

We apply three different noise addition techniques, which we call *Random Perturbation Technique (RPT)*, *Probabilistic Perturbation Technique (PPT)* and *All Leaves Probabilistic Perturbation Technique (ALPT)*. In each of these techniques the same amount of noise is added, where the amount of noise is measured by the expected number of changed classes $E(N)$ in the perturbed data set, and

$$E(N) = \sum_{k=1}^{H} \frac{2m_k n_k}{m_k + n_k}. \tag{4.1}$$

We introduce these three techniques as follows, where we consider a data set having the class attribute of domain size two. However, these techniques can be generalized for data sets having class attributes with bigger domain sizes.

Random Perturbation Technique

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 the 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 the minority class. This change is made in all heterogeneous leaves.

The RPT technique always results in an even number of values being modified. For example, if there is only one minority record then total number of changed values can be either zero or two. If we choose the same record twice (once while changing the minority class to the majority class and the other time while changing the majority class to the minority class) then the number of changed values will be zero. Otherwise if we choose two different records while changing the class values then the number of changed values will be two.

Let, p_k^{2i} be the probability that $2i$ class attribute values of the kth heterogeneous leaf will be changed from their original values. Then,

$$p_k^{2i} = \frac{\binom{n_k}{i}\binom{m_k}{i}}{\binom{m_k+n_k}{n_k}};$$

and the expected number of changed classes in the kth heterogeneous leaf is,

$$\begin{aligned} E(N_k) &= \sum_{i=0}^{n_k}(2i) \times p_k^{2i} \\ &= 2\sum_{i=1}^{n_k} i\frac{\binom{n_k}{i}\binom{m_k}{i}}{\binom{m_k+n_k}{n_k}} \\ &= \frac{2}{\binom{m_k+n_k}{n_k}}\sum_{i=1}^{n_k} i\binom{n_k}{i}\binom{m_k}{i}; \end{aligned}$$

Since,

$$\begin{aligned} \binom{n}{i} &= \frac{n(n-1)!}{i(i-1)!(n-i)!} \\ &= \frac{n}{i}\binom{n-1}{i-1}; \end{aligned}$$

we have

$$\begin{aligned} E(N_k) &= \frac{2}{\binom{m_k+n_k}{n_k}}\sum_{i=1}^{n_k} i\binom{n_k}{i}\frac{m_k}{i}\binom{m_k-1}{i-1} \\ &= \frac{2m_k}{\binom{m_k+n_k}{n_k}}\sum_{i=1}^{n_k}\binom{n_k}{i}\binom{m_k-1}{i-1}; \end{aligned}$$

Assume, $l_k = n_k - i$, then $i = n_k - l_k$ and $l_k \in [0, n_k - 1]$.

Therefore,

$$E(N_k) = \frac{2m_k}{\binom{m_k+n_k}{n_k}} \sum_{l_k=0}^{n_k-1} \binom{n_k}{n_k - l_k} \binom{m_k - 1}{n_k - l_k - 1};$$

Since,

$$\binom{n}{i} = \binom{n}{n-i};$$

we have

$$\begin{aligned} E(N_k) &= \frac{2m_k}{\binom{m_k+n_k}{n_k}} \sum_{l_k=0}^{n_k-1} \binom{n_k}{l_k} \binom{m_k - 1}{n_k - l_k - 1} \\ &= \frac{2m_k}{\binom{m_k+n_k}{n_k}} \binom{n_k + m_k - 1}{n_k - 1} \\ &= \frac{2m_k \binom{m_k+n_k-1}{n_k-1}}{\frac{m_k+n_k}{n_k}\binom{m_k+n_k-1}{n_k-1}} \\ &= \frac{2m_k n_k}{m_k + n_k}; \end{aligned}$$

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

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

However, the probability that a class has been perturbed is not uniformly distributed over all the records in the heterogeneous leaves. The records that have minority class in the perturbed data set are perturbed with the probability $\frac{m_k}{m_k+n_k}$, while the records with majority class in the perturbed data set are perturbed with the probability $\frac{n_k}{m_k+n_k}$. An intruder's best strategy is to assume that the records in the k-th heterogeneous leaf belong to the majority class with probability $\frac{m_k}{m_k+n_k}$. In other words, an intruder has no way of identifying records, which originally belonged to the minority class. Thus, the security of these records is very high. On the other hand, security of records that belong to the majority class is very low, while the security of records that belong to homogeneous leaves is zero.

Probabilistic Perturbation Technique

In *Probabilistic Perturbation Technique (PPT)*, 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 the class of all records in the kth heterogeneous leaf are changed to minority class with a probability $p_k = \frac{n_k}{m_k+n_k}$. Therefore, the expected number of changed classes in the kth heterogeneous leaf of the perturbed data set is:

$$\begin{aligned} E(N_k) &= m_k p_k + n_k(1-p_k) \\ &= n_k + p_k(m_k - n_k) \\ &= n_k + \frac{n_k}{m_k+n_k}(m_k - n_k) \\ &= \frac{2m_k n_k}{m_k+n_k}. \end{aligned}$$

The expected number of changed classes in the whole perturbed data set is,

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

Although the expected number of changed classes is the same as in the Random technique, the security is slightly higher as the intruder does not know the exact probability that a given record belongs to the majority class. For the leaf k, this probability is drawn from the binomial distribution with the mean $\mu = \frac{2m_k n_k}{m_k+n_k}$. As we shall see in the next section, our experiments indicate that the data quality of a data set perturbed by *PPT* is also slightly worse than the data quality of the data set perturbed by the *RPT*.

All Leaves Probabilistic Technique

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 our other techniques to this one in order to evaluate the effectiveness the other techniques in pattern preservation.

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,

$$\begin{aligned} E(N) &= \sum_{i=1}^{N_{Total}} p \\ &= N_{Total} \times \frac{1}{N_{Total}} \sum_{k=1}^{H} \frac{2m_k n_k}{m_k + n_k} \\ &= \sum_{k=1}^{H} \frac{2m_k n_k}{m_k + n_k}. \end{aligned}$$

We measure security again by the probability that a class value in the perturbed file is not the same as the corresponding value in the original file. This probability is now uniformly distributed over all records in the data set and is equal to $p = \frac{1}{N_{Total}} \sum_{k}^{H} \frac{2m_k n_k}{m_k + n_k}$.

We recall that the security of records in homogeneous leaves in the previous two techniques (RPT and PPT) is zero while the security of records with minority class in the original data set is very high. Therefore, the security levels of the records are not uniformly distributed in both RPT and PPT. Following the arguments previously given in this chapter regarding the importance of the security for minority records over the security for records that belong to a homogeneous leaf, we consider the nonuniform distribution more favorable than the uniform distribution provided by $ALPT$. We also note that the total number of records having a particular class remains the same in the perturbed data set when we use the RPT. This is not the case with PPT and $ALPT$.

Generalization

We next show how these three techniques can be generalized for data sets having the class attribute of domain size greater than two. We use the following notation.

H - number of heterogeneous leaves

c_k^m - the majority class in the kth heterogeneous leaf

m_k - number of majority records in the kth heterogeneous leaf; $1 \leq k \leq H$

c_k^1, c_k^2, ... c_k^p - minority classes in the kth heterogeneous leaf; $1 \leq p \leq (d-1)$, where d is the domain size of the class attribute

n_k^i - number of minority records corresponding to the minority class c_k^i, where $1 \le i \le p$

n_k - total number of minority records in the kth heterogeneous leaf, $n_k = \Sigma_{i=1}^{p} n_k^i$

In *RPT* we first convert the class values of all n_k minority records to the majority class c_k^m, in the kth heterogeneous leaf. We next choose any n_k^1 number of records randomly from the set of records of the leaf, and convert the class value from c_k^m to c_k^1. Then, from the rest of the records belonging to the leaf, we choose any n_k^2 number of records randomly, and convert the class value to c_k^2. We repeat the process for all minority classes of the leaf. The change is made in all heterogeneous leaves.

In *PPT* the class values of all n_k minority records are converted to the majority class c_k^m, in the kth heterogeneous leaf. Then the class of all records belonging to the leaf are changed to c_k^i minority class with a probability $p_k^i = \frac{n_k^i}{m_k+n_k}$, where $1 \le i \le p$.

In *ALPT* we change the class of all records with a 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. If the the original class value c_o of a record is changed, then it is changed to a class value c_p with a probability $p_p = \frac{R_p}{N_{Total}-R_o}$, where R_p and R_o are the numbers of records with the class value c_p and c_o, respectively, and $1 \le p \le (d-1)$.

4.3 The Experiment

The purpose of this experiment is to evaluate the effectiveness of our perturbation techniques, namely *RPT* and *PPT* in preserving the original patterns in the perturbed data sets. We compare these techniques with the *ALPT* which we use as a simulation of natural noise occurring in the class attribute. We add the same amount of noise in all of these techniques.

We perturb a data set by all three techniques, and thereby produce three different perturbed data sets. We build decision trees from the original data set (original tree) and all perturbed data sets (perturbed trees). The similarities of these perturbed trees with the original tree are evaluated and compared. The similarity of a perturbed tree with an original tree is evaluated based on various criteria such as the classification rules, attributes tested, and number of records belonging to each classification rule. This evaluation and comparison help us to decide which technique preserves the original patterns the best.

If the number of classification errors of a tree (on the data set from which it is built) is low, then the tree represents the patterns of the data set well. If two good representative trees are found to be similar, then the two underlying data sets (from which the trees are built) are also similar in terms of the patterns discovered by the trees. Therefore, if a perturbed tree represents the perturbed data set well, and is similar to the original tree then the perturbed data set preserves the original patterns well. We use this concept in evaluating the data quality of a perturbed data set.

Data mining generally extracts information from data sets having a huge number of records. Therefore, huge data sets are used to test the efficiency and correctness of a new data mining method such as a new classification and a new clustering technique. However, the main purpose of our noise addition technique is protecting privacy in data mining - instead of performing a data mining task. We protect privacy by hiding sensitive information, and preserve the patterns by disturbing the data set the least. It is not supposed to be difficult to hide sensitive information and preserve patterns in a big and dense data set, if we can do that in a small data set. Therefore, we do not necessarily need to use huge data sets for our experiments. Our initial experiments in this chapter, Chapter 5 and Chapter 6 use small data sets. However, we use bigger data sets for our main experiments in Chapter 7.

In this section we present some experimental results on the *Boston Housing Price (BHP)* data set with 300 records. The data set is widely used by the data mining community and is available from the UCI Machine Learning Repository [77]. We use Quinlan's famous decision tree builder See5 (commercially available) produced by RuleQuest Research.

The *BHP* Data Set

The *Boston Housing Price (BHP)* data set has altogether 12 attributes, out of which one is the categorical class attribute with domain {top 20%, bottom 80% }. The non-class attributes are crime rate, proportion large lots, proportion industrial, nitric oxides ppm, av rooms per dwelling, proportion pre-1940, distance to employment centers, accessibility to radial highways, property tax rate per 10,000 dollars, pupil-teacher ratio and percentage low income earners. All non-class attributes are continuous. We ignore two other non-class attributes "CHAS" and "B" throughout our experiments. We first build a decision tree from the 300 records of original *BHP* data set, which is shown in Figure 4.2.

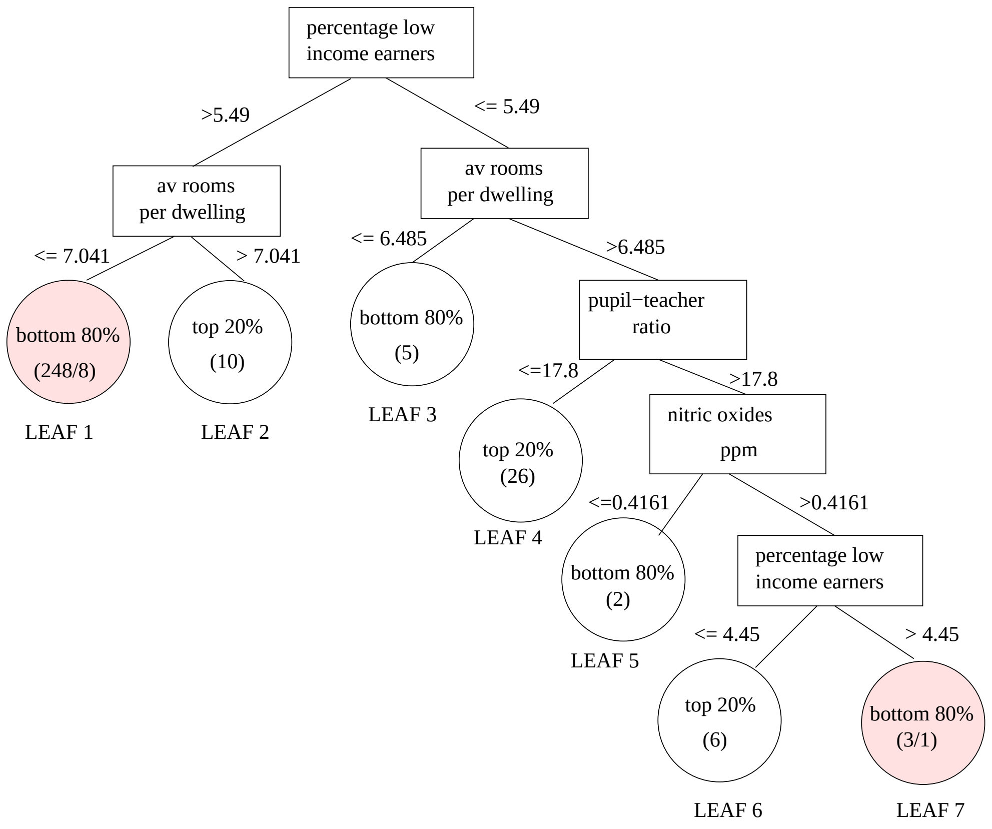

Figure 4.2: The decision tree obtained from 300 records of the original *BHP* data set.

Results

We perturb the data set 5 times by the *RPT*, and thereby produce 5 perturbed data sets. We build a decision tree from each of these perturbed data sets. The decision trees are shown in the figures from Figure 4.3 to Figure 4.7. We next perturb the original *BHP* data set 10 times by the *PPT* and thereby, produce 10 perturbed data sets. We build a decision tree from each of these perturbed data sets. A few of these decision trees are shown in the figures from Figure 4.8 to Figure 4.10. Finally, we perturb the original *BHP* data set 10 times by the *ALPT*. We build a decision tree from each of these perturbed data sets. A few of these decision trees are shown in the figures from Figure 4.11 to Figure 4.13.

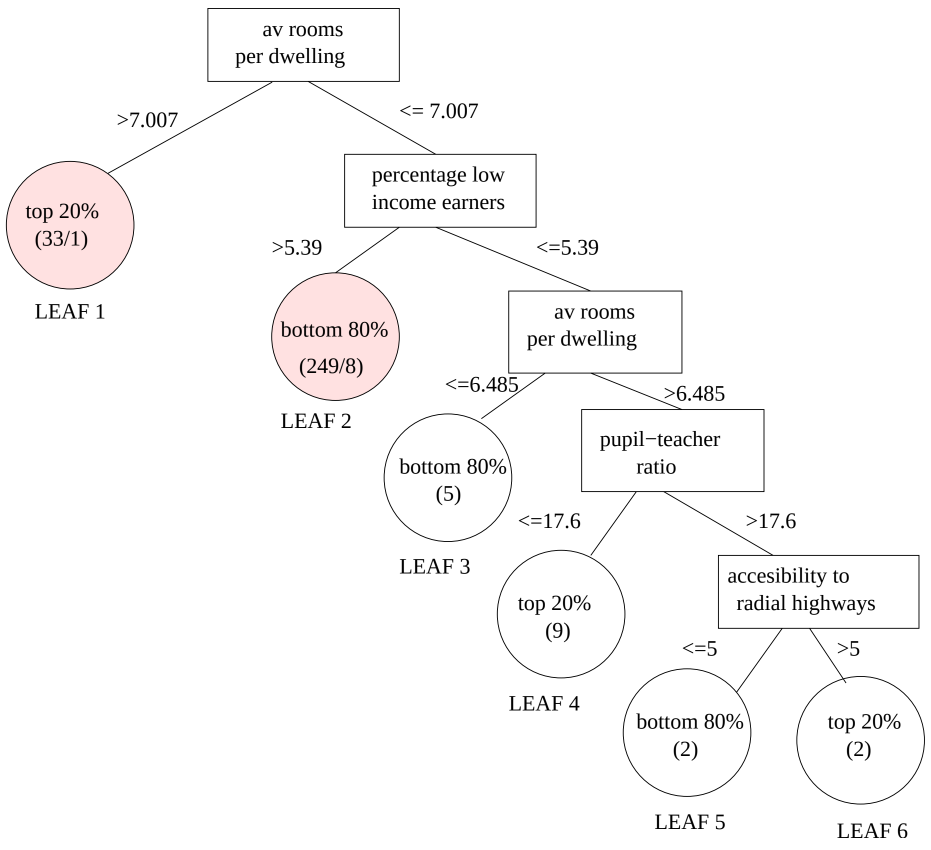

Figure 4.3: The decision tree obtained from the 1st of the five *BHP* data sets that have been perturbed by the *RPT*.

Result Analysis

After careful analysis we find that the data sets perturbed by *RPT* or *PPT*, generally preserve the original patterns better than the data sets perturbed by *ALPT*. However, between *RPT* and *PPT*, the first one is more consistent in preserving the patterns.

The classification rule for Leaf 1 of the original tree (Figure 4.2) is *percentage low income earners*$>$5.49 & *av rooms per dwelling*$<=$ 7.041 $\Rightarrow$ *bottom 80%*. This classification rule applies to 248 records, out of 300 records of the data set. If we carefully analyze the trees obtained from data sets perturbed by *RPT* (shown in figures from Figure 4.3 to Figure 4.7), we find that in all of these trees the rule is preserved with some slight changes in the splitting points. For example, in Figure 4.3 the rule for the Leaf 2 is *percentage low income earners*$>$5.39 & *av rooms per dwelling*$<=$ 7.007 $\Rightarrow$ *bottom 80%*. This rule also relates to 249 records of the data set.

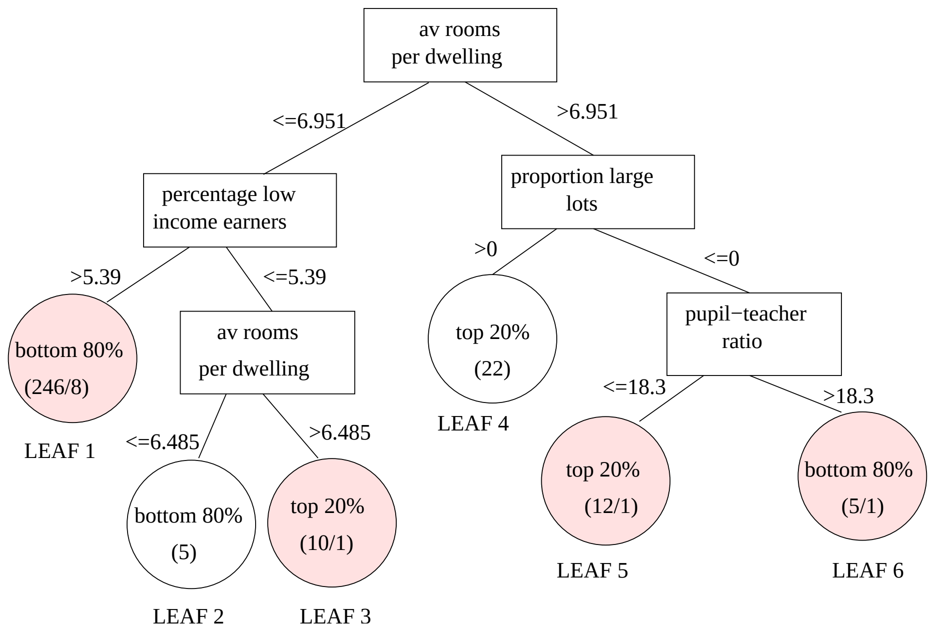

Figure 4.4: The decision tree obtained from the 2nd of the five BHP data sets that have been perturbed by the RPT.

The same rule is also preserved in PPT perturbed trees shown in Figure 4.9 (see the rule for Leaf 1) and Figure 4.10 (see the rule for Leaf 4). However, the rule is not preserved as it is in the PPT perturbed tree shown in Figure 4.8 (see the rule for Leaf 1). In 9 out of our 10 experiments this rule is preserved in the PPT perturbed trees. The rule is also preserved in the $ALPT$ perturbed tree shown in Figure 4.12 (see the rule for Leaf 2). However, the tree has 20 numbers of misclassified records (errors) on the underlying data set - while the original tree (shown in Figure 4.2) has only 9 errors on the original data set.

The same $ALPT$ perturbed tree also has 19 errors for the classification rule (under discussion) on the $ALPT$ perturbed data set, compared to 8 errors made by the original tree for the rule on the original data set. This suggests that although the rule is preserved in the perturbed tree, the underlying data set does not match with the rule as good as the original data set does. Therefore, just the preservation of the rule in the perturbed tree does not indicate good data quality of the perturbed data set. Moreover, the other two $ALPT$ perturbed trees shown in Figure 4.11 and Figure 4.13 do not even preserve the rule, in its original form. In 5 out of total 10 experiments, the perturbed trees do not preserve the rule. The trees that preserve the rule have errors between 20 to 27 on their corresponding

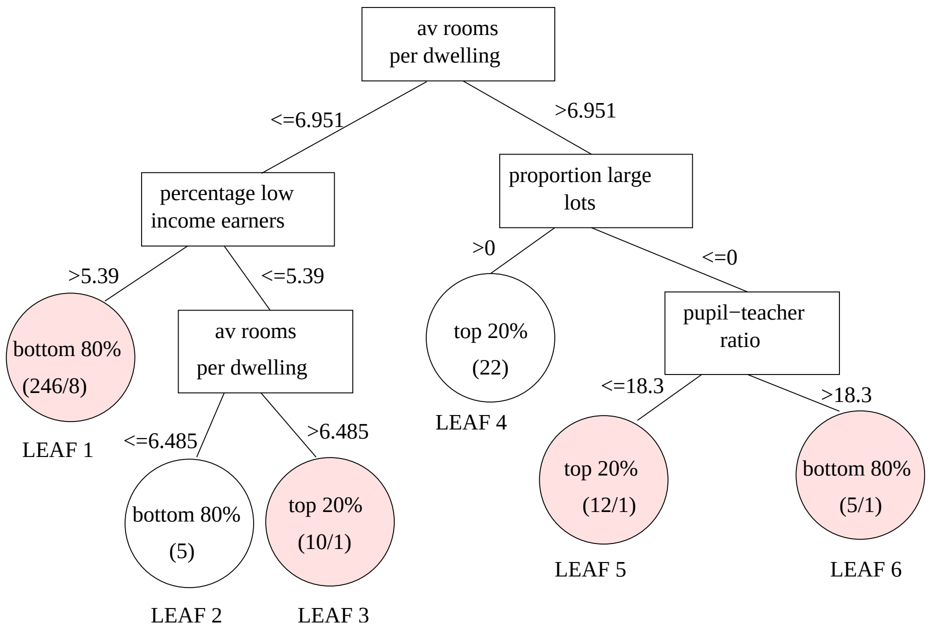

Figure 4.5: The decision tree obtained from the 3rd of the five BHP data sets that have been perturbed by the RPT.

underlying data sets.

Similarly, there are other classification rules such as the rule for the Leaf 3 and the rule for the Leaf 4 of the original tree (Figure 4.2), that are preserved in all or most of the RPT perturbed trees. Some of these rules are also preserved in many PPT perturbed trees.

The number of misclassified records for the original tree on the original data set is 9, whereas the number of misclassified records for the RPT perturbed trees on their corresponding underlying data sets varies from 9 to 11. Similarly, number of misclassified records for PPT varies from 6 to 13, whereas for $ALPT$ it varies from 17 to 27.

Trees built from the RPT perturbed data sets are very similar to the tree produced from the original data set. Out of four attributes tested in the original tree, three appear in all RPT perturbed trees. The splitting points used for these three attributes, in all trees, are very similar to the corresponding splitting points used in the original tree. For example, the splitting point of *percentage low income earners* in the original tree is 5.49 - whereas it ranges from 5.39 to 5.49 in all RPT perturbed trees. The attribute from the original tree, which is missing in the perturbed trees, is only tested towards the bottom of the original tree and caters for only 11 records out of 300. Ignoring logic rules including that attribute, four other rules from the original tree appear in the same or very similar form in all trees.

The analysis of the ten *PPT* perturbed trees shows results similar to the results obtained from the analysis of *RPT* perturbed trees. Out of 4 attributes from the original tree 2 appear in all perturbed trees, while the third one appears in 8 out of 10 trees. The logic rules from the original tree (with the exception of the rule involving *nitric oxides ppm*) appear in the same or very similar form in most trees. However, 4 trees contain new rules and have significant number of cases belonging to those new rules.

Trees obtained from the 10 *ALPT* perturbed data sets are found significantly different from the original tree. Still, all of them contain the two most significant attributes from the original tree and 7 out of 10 trees contain the third attribute. Some of these trees are much deeper than the original tree and contain quite a few new rules. On the other hand some of the trees are very shallow and test only two attributes.

4.4 Conclusion

In this chapter, following [28], we have carefully added a little amount of noise to the confidential class attribute values of a data set for preserving individual privacy. We have added noise to the class values using two of our noise addition techniques namely *RPT* and *PPT*. We have compared the data sets perturbed by these two techniques with the data sets perturbed by another technique called *ALPT* through studying the similarities between the decision trees obtained from these perturbed data sets and the original data set. Although we have added the same amount of noise in all three techniques - our experimental results show that the first two techniques preserve the patterns far better than the third technique, which we have used as a simulation of natural noise occurring in the class values. Between the two techniques, *PPT* provides better security, while *RPT* preserves the patterns better. In the next chapter we present novel techniques for adding noise to all non-class numerical attributes.

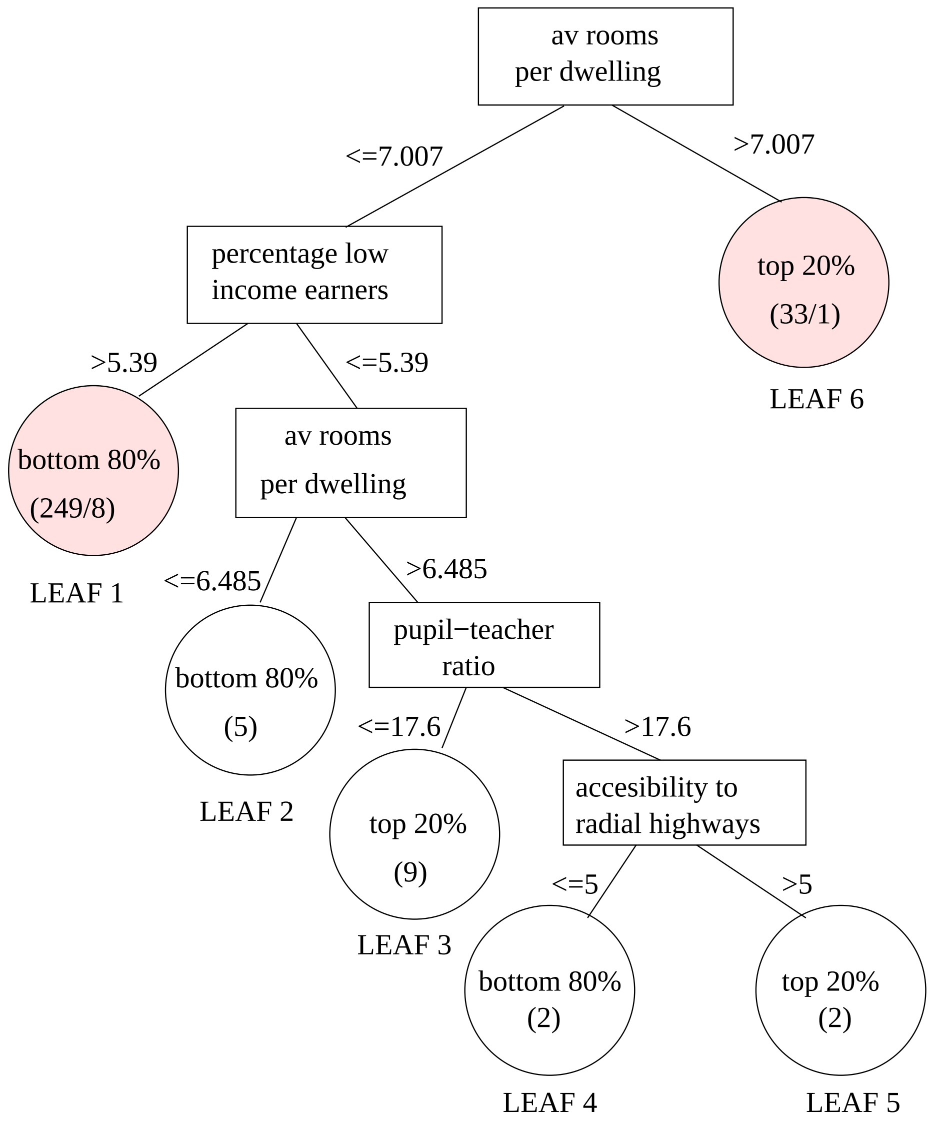

Figure 4.6: The decision tree obtained from the 4th of the five *BHP* data sets that have been perturbed by the *RPT*.

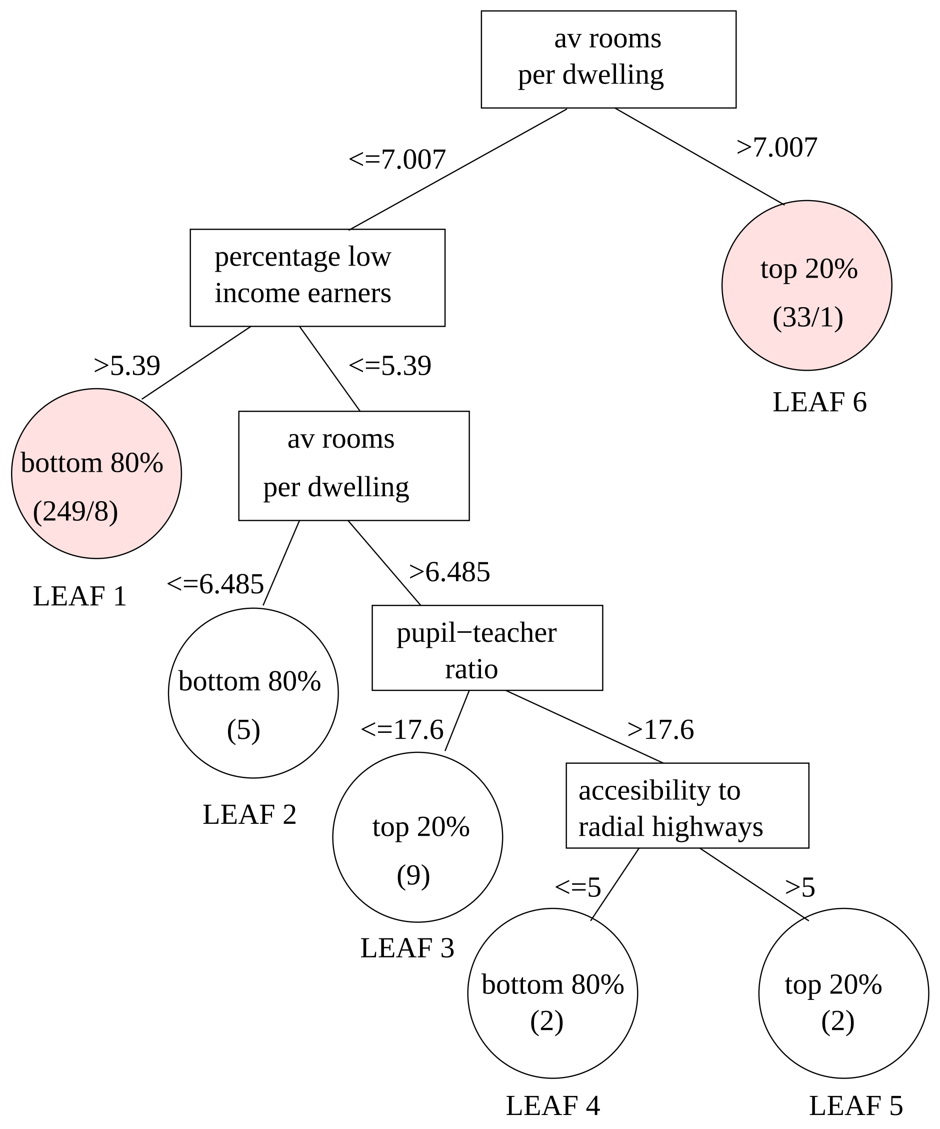

Figure 4.7: The decision tree obtained from the 5th of the five *BHP* data sets that have been perturbed by the *RPT*.

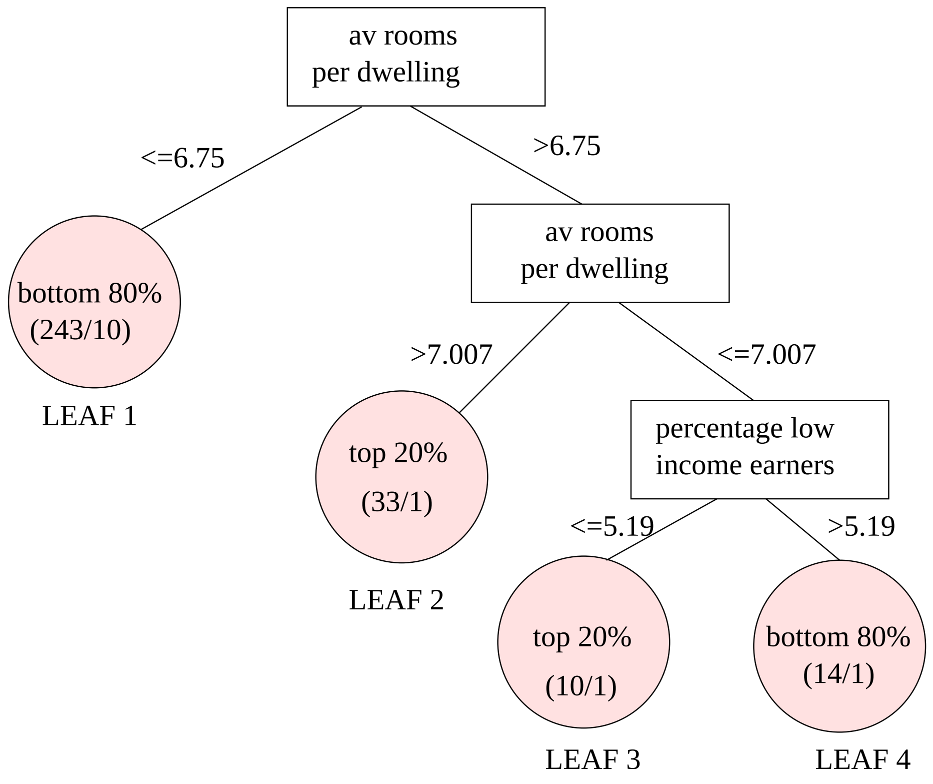

Figure 4.8: The decision tree obtained from one of the ten *BHP* data sets that have been perturbed by the *PPT*.

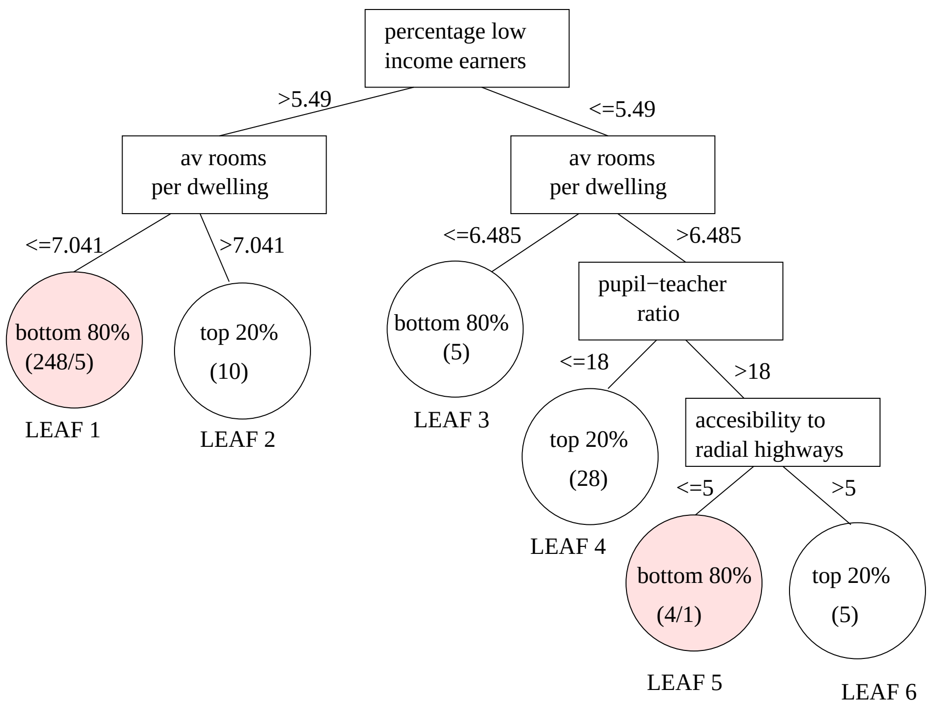

Figure 4.9: The decision tree obtained from another *BHP* data set that has been perturbed by the *PPT*.

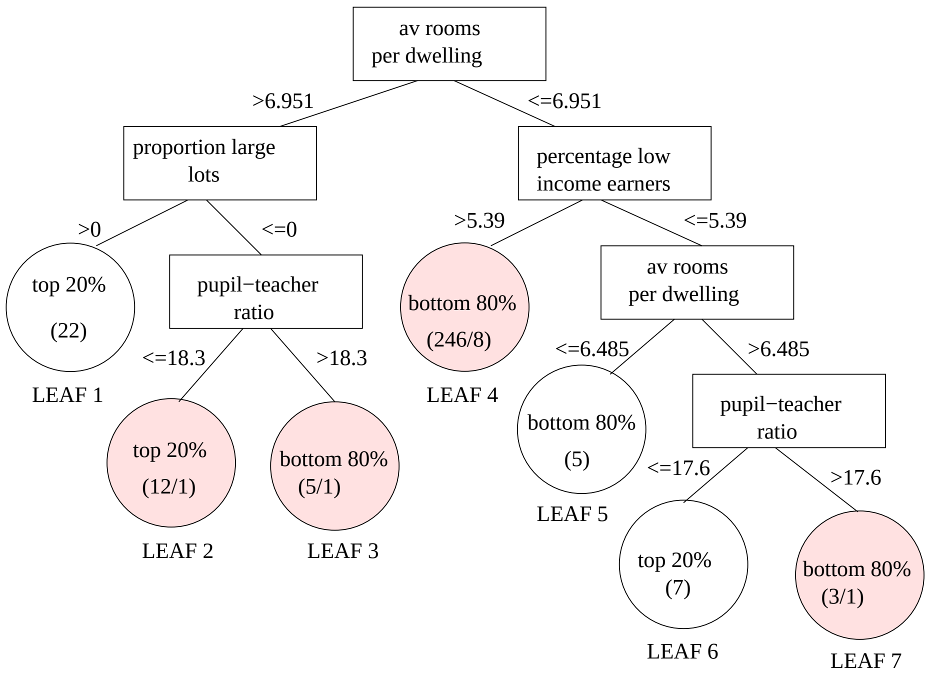

Figure 4.10: The decision tree obtained from a 3rd *BHP* data set that has been perturbed by the *PPT*.

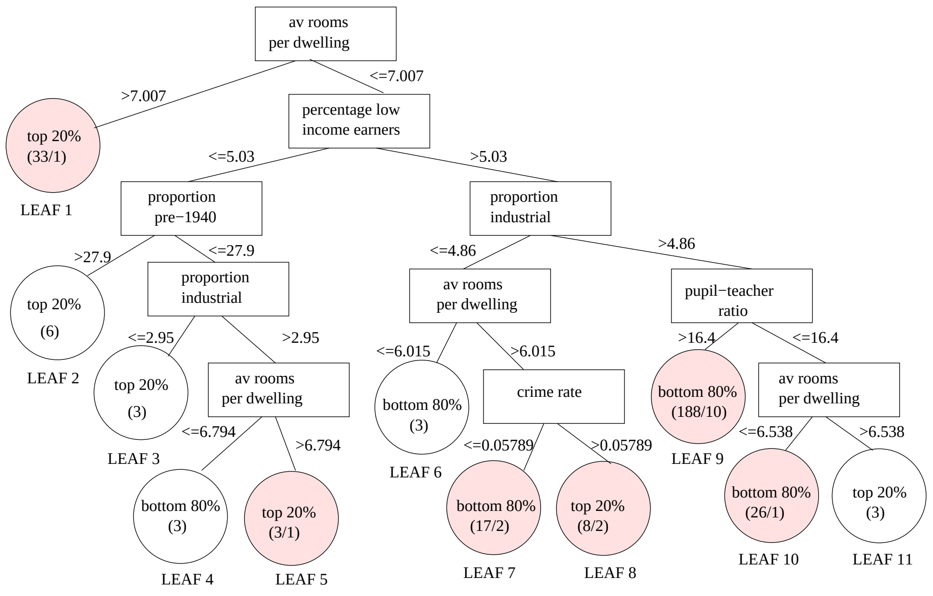

Figure 4.11: The decision tree obtained from one of the ten *BHP* data sets that have been perturbed by the *ALPT*.

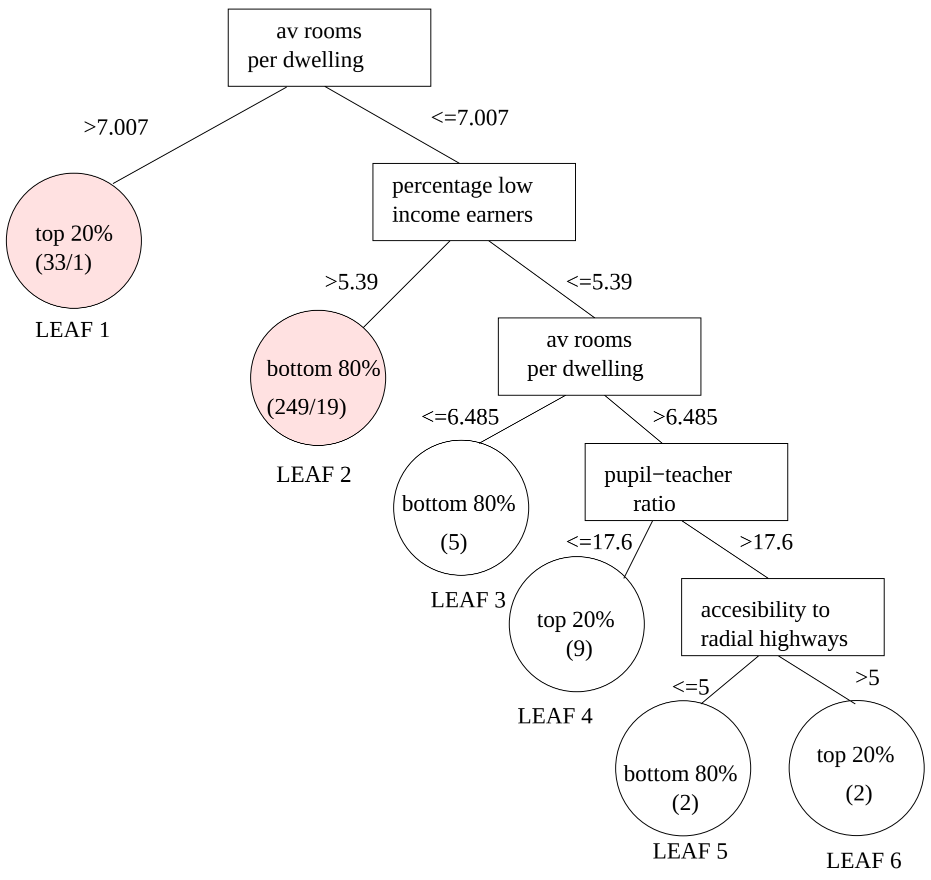

Figure 4.12: The decision tree obtained from another *BHP* data set that has been perturbed by the *ALPT*.

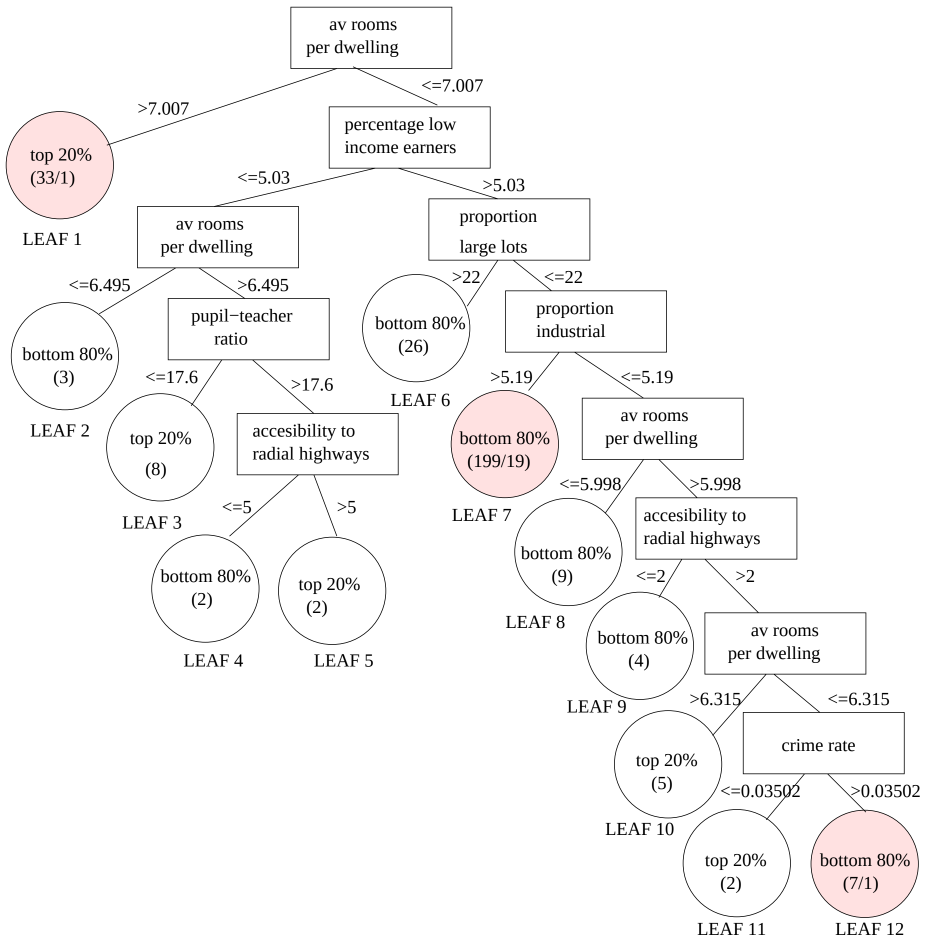

Figure 4.13: The decision tree obtained from a 3rd *BHP* data set that has been perturbed by the *ALPT*.

Chapter 5

Non-class Numerical Attributes Perturbation Technique

5.1 Introduction

We continue with our objectives of protecting the confidentiality of the class value belonging to an individual, while maintaining a good data quality in the released data set. In the previous chapter, following the technique proposed by Estivill-Castro and Brankovic [28], we have presented a couple of new noise addition techniques for perturbing sensitive class attribute values. In this chapter we present a couple of techniques for adding noise to non-class numerical attributes. The technique proposed by Estivill-Castro and Brankovic [28] adds noise to categorical class attribute only.

Noise addition to the class attribute reduces the disclosure risk of the confidential class value belonging to an individual. Additionally, noise addition to the non-class attributes along with the class attribute results in even better privacy. Moreover, some non-class numerical attributes can also be considered confidential in the sense that disclosure of such attribute values belonging to an individual can expose personal information to an unacceptable level. For example, numerical non-class attributes such as "*Salary*", "*Credit Card Limit*", "*Home Equity*" and "*Liabilities*" can be considered confidential. Some other numerical attributes such as "*Height*" and "*Country of Origin*" can be considered non confidential, but again the degree of confidentiality of an attribute is context dependant. Some noise addition techniques such as the one proposed by Muralidhar et al. [73] add noise only to confidential attributes. However, we suggest noise addition to all of the attributes regardless of whether or not they are considered confidential. In this chapter we present a noise addition technique for all non-class numerical attributes of a data set having several numerical attributes and a single categorical class attribute. An example of such a data set is the *Wisconsin Breast Cancer (WBC)* data set available from the UCI Machine Learning Repository [77]. We first give a brief introduction to the *WBC* data set, as we shall refer to it in examples throughout this chapter.

The *WBC* data set has 10 numerical non-class attributes and one categorical class attribute. The class attribute has two categorical values, "2" and "4". Out of 10 numerical attributes one is the *Record ID*, which uniquely identifies a record. Therefore, this attribute is excluded from the data set at the beginning. The remaining 9 numerical non-class attributes are *Clump Thickness*, *Uniformity of Cell Size*, *Uniformity of Cell Shape*, *Marginal Adhesion*, *Single Epithelial Cell Size*, *Bare Nuclei*, *Bland Chromatin*, *Normal Nucleoli* and *Mitoses.* The domain of each of these attributes is [1,10], where the domain of an attribute is a set of legal values that the attribute can take [19]. We build a decision tree, from 349 records of the *WBC* data set, using the commercially available of the See5 decision tree builder of RuleQuest Reseasrch. The decision tree is shown in the Figure 5.1.

In order to add noise to non-class numerical attributes we split the non-class attributes of each record into two divisions, namely *Leaf Innocent Attributes (LINNAs)* and *Leaf Influential Attributes (LINFAs).* In a decision tree, if an attribute is not tested in any of the nodes on the path between the root and a leaf, then the attribute is called a *Leaf Innocent Attribute (LINNA)* for the records belonging to that leaf. Thus, records belonging to every leaf of a decision tree maintain a set of *LINNAs.* For example, *Uniformity of Cell Shape*, *Marginal Adhesion*, *Single Epithelial Cell Size*, *Bare Nuclei*, *Bland Chromatin* and *Mitoses* are the *LINNAs* for the records belonging to Leaf 1 of the decision tree shown in Figure 5.1. *LINNAs* are not influential in the sense that they do not play any role in the prediction of the class attribute for the records belonging to the leaf. In other words, they do not appear in the logic rule related to the leaf.

On the contrary, the attribute that is tested at least once on the path between the root and a leaf is called a *Leaf Influential Attribute (LINFA)* for the records belonging to that particular leaf. For each leaf of a decision tree there is a set of *LINFAs.* For example, *Clump Thickness*, *Uniformity of Cell Size* and *Normal Nucleoli* are the *LINFAs* for the records belonging to the Leaf 1. Conceptually the selection of *LINFAs* and *LINNAs* is similar

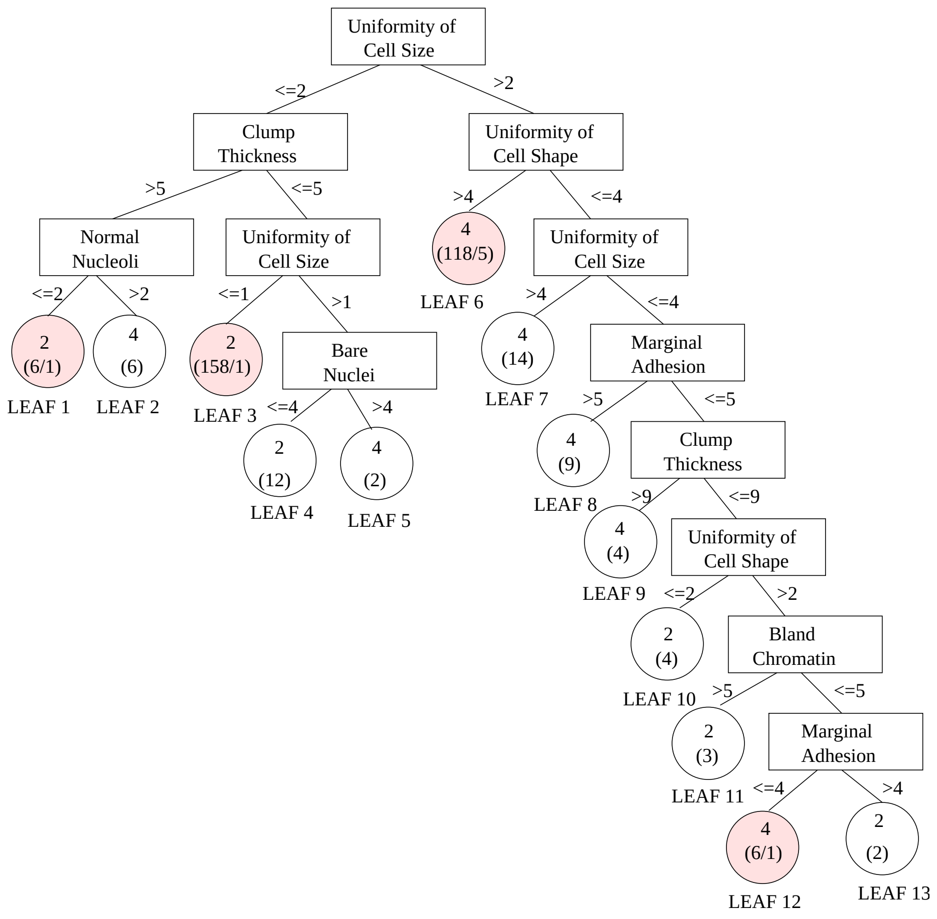

Figure 5.1: The decision tree obtained from 349 records of the original (unperturbed) *WBC* data set.

to feature selection in the sense that a feature selection algorithm would also extract the *LINFAs* since they are the truly informative attributes. However, instead of having a set of *LINFAs* for whole data set we have separate sets of *LINFAs* for leaves.

We add noise to the *LINNAs* of the records through a novel noise addition technique called *Leaf Innocent Attribute Perturbation Technique (LINNAPT)*. We also add noise to the *LINFAs* of the records by another novel noise addition technique called *Leaf Influential Attribute Perturbation Technique (LINFAPT)*. *LINNAPT* adds noise only to the *LINNAs*, whereas *LINFAPT* adds noise only to the *LINFAs*. These techniques are described in the following sections.

Therefore, in order to add noise to both *LINNAs* and *LINFAs* we first build a decision tree from an unperturbed data set. The set of *LINNAs* and the set of *LINFAs* are determined for the records belonging to a leaf. The *LINNAs* are first perturbed by *LINNAPT* and then the *LINFAs* are perturbed by *LINFAPT*. Eventually the records belonging to all leaves are perturbed by these two techniques. The effectiveness of these techniques in maintaining the data quality is evaluated through comparing it with another noise addition technique called *Random Noise Addition Technique (RNAT)*, which is not catered for preserving the patterns unlike the other techniques. We describe the *LINNAPT*, the *LINFAPT* and *RNAT* in the following sections.

5.2 The *Leaf Innocent Attribute Perturbation Technique*

Let $\mathbf{A}$ be a set of all *LINNAs*. The *LINNAPT* adds noise to $\mathbf{A}$ and produces the set of perturbed attributes $\mathbf{A}^* = \mathbf{A} + \xi$, where ξ is discrete noise with a mean μ and a variance σ^2. The distribution of the noise can be chosen to suit a particular application. The domains of $\mathbf{A}^*$ remain the same as the domain of $\mathbf{A}$. For example, the domain of a perturbed attribute $A^* \in \mathbf{A}^*$ remains the same as the domain of the corresponding unperturbed attribute $A \in \mathbf{A}$. We take a wrap around approach to preserve the domain of an attribute. *LINNAPT* adds noise to all *LINNAs* of each and every record of the data set. We present a pseudocode as follows.

```
For each record, DO:
    STEP 1: Determine the leaf $L$ that the record belongs to.
    STEP 2: From the original decision tree compute the list of LINNAs for $L$.
    STEP 3: Compute domains of the LINNAs.
    STEP 4: For each LINNA, $A \in \mathbf{A}$, add noise to produce a perturbed attribute $A^* = A + \xi$, where $\xi$ is drawn from a normal distribution having mean $\mu$ and a variance $\sigma^2$.
    STEP 5: If a value of $A^*$ falls outside the domain of $A$ wrap around the value so that it remains within the domain of $A$.
END DO
```

We again evaluate the data quality of a data set perturbed by *LINNAPT* through the similarity of the decision trees obtained from the perturbed data set and the original data set. In Chapter 7 we present some experimental results to evaluate the data quality of a *LINNAPT* perturbed data set. We discuss the techniques and their overall security in Chapter 8 in detail. We also present some preliminary security analyses of a data set perturbed only by *LINNAPT* as follows.

To evaluate the security of our method, we consider the following scenario. Assume that an intruder is interested in learning the confidential class of a record X in a data set. We assume that the intruder has some supplementary knowledge about the record X, that is, he/she knows the values belonging to few attributes of the record X, as otherwise he/she would not be able to identify the record and learn the class.

We shall first assume that an intruder can uniquely re-identify the record X with the supplementary knowledge of the values of *LINNAs*. After the noise has been added to the *LINNAs* the intruder is not able to uniquely re-identify the record any more. They can, however, estimate the probability $p(X \rightarrow Y)$ that a record X in the original data set is changed to the record Y in the perturbed data set, assuming of course that the probability distribution of the added noise is known to them. We can express $p(X \rightarrow Y)$ as follows:

$$p(X \rightarrow Y) \quad = \quad \prod_{i=1}^{k} p(A_i^* - A_i),$$

where $p(A_i^* - A_i)$ is the probability that the noise added to the attribute A_i is equal to $(A_i^* - A_i)$, and k is the number of *LINNAs* required to re-identify the record X. If the data set is dense, that is, if there are many records with similar values in *LINNAs* then there will be many records Y_i in the perturb file with a similar probability $p(X \rightarrow Y_i)$. In general, these records will belong to different leaves with different leaf classes and intruder will have a great deal of uncertainty about the class of X. For a record X of the original data set, if the distribution of $p(X \rightarrow Y)$ over all perturbed records is uniform then the security is high. In fact, the more uniform the distribution, the higher the security.

We shall next assume that an intruder can uniquely identify the record X with the supplementary knowledge about the values of *LINFAs*. The intruder will still be able to re-identify the record X, since no noise has been added to the *LINFAs*, and thus the intruder can learn the class value of X. We, however, argue that in this case the records from the released data set are not under greater privacy threat than any other record of that kind which is not in the released data set. Indeed, the intruder can obtain a pretty good estimate of the class value of any record, for which he knows the values of *LINFAs*, through the classifier built from the released data set. However, one can argue that classifiers are typically more accurate when applied to the records from a training set (released data set, in our scenario) than when applied to other records that are not in the training set.

Therefore, we add noise to the *LINFAs* on top of the noise addition to the *LINNAs* and the class attribute.

5.3 The *Leaf Influential Attribute Perturbation Technique*

Let $\mathbf{B}$ be a set of all *LINFAs*. The *LINFAPT* adds noise to $\mathbf{B}$ and produces the set of perturbed attributes $\mathbf{B}^* = \mathbf{B} + \xi$, where ξ is random noise generated from a normal distribution with a mean μ and a variance σ^2. The distribution of the noise can be chosen to suit a particular application. The domains of $\mathbf{B}^*$ remain the same as the domains of $\mathbf{B}$. For example, the domain of a perturbed attribute $B^* \in \mathbf{B}^*$ remains the same as the domain of the corresponding unperturbed attribute $B \in \mathbf{B}$. We take a wrap around approach to preserve the domain of an attribute. *LINFAPT* adds noise to all *LINFAs* of each and every record of a data set. We present a pseudocode as follows.

```
For each record, DO:
    STEP 1: Determine the leaf $L$ that the record belongs to.
    STEP 2: From the original decision tree compute the list of LINFAs (say, $\mathbf{B}$) for $L$.
    STEP 3: From the original decision tree compute the domains of the LINFAs. Domain of a LINFA (say, $B \in \mathbf{B}$) is the range defined by the conditional values of the LINFA for $L$.
    STEP 4: For each LINFA, $B \in \mathbf{B}$, add noise to produce a perturbed attribute $B^* = B + \xi$, where $\xi$ is drawn from a normal distribution having mean $\mu$ and a variance $\sigma^2$.
    STEP 5: If a value of $B^*$ falls outside the domain of $B$ wrap around the value so it remains within the domain of $B$.
END DO
```

An important characteristic of *LINFAPT* is that the perturbed values of all *LINFAs* remain within the ranges defined by the conditional values of the *LINFAs* in a decision tree. *LINFAPT* uses a wrap around approach when a perturbed value B falls outside the range. If B is greater than the upper limit u of the range $r = [l, u]$ then a final perturbed value is calculated as $B_f = l + B - u - 1$. A similar approach is taken if B is less than l.

As an example, consider Leaf 1 of Figure 5.1 that has three *LINFAs* namely *Uniformity of Cell Size*, *Clump Thickness* and *Normal Nucleoli.* The range of *Uniformity of Cell Size*, defined by the conditional value of the *LINFA* for Leaf 1 is 1 to 2 inclusive. Thus, *LINFAPT* adds noise to this attribute for the records belonging to Leaf 1 in such a way so that the perturbed value of the attribute remains within the range 1 to 2. Similarly, the range of

Clump Thickness for the records belonging to Leaf 1 is 6 to 10 inclusive, while the range of *Normal Nucleoli* for the records belonging to the same leaf is 1 to 2 inclusive. The range of *Uniformity of Cell Size* for the records belonging to Leaf 7 is 5 to 10 inclusive.

We continue with our usual approach of measuring the data quality of a perturbed data set through the similarity analysis of the decision trees obtained from the original and the perturbed data sets. Experimental results are presented in Chapter 7 to evaluate the data quality of a perturbed data set. The security provided by *LINFAPT* can be evaluated in a way similar to the evaluation of the security provided by *LINNAPT*. Moreover, an overall security analyses of a released data set is discussed in Chapter 8.

5.4 The *Random Noise Addition Technique*

We now present another noise addition technique called *Random Noise Addition Technique (RNAT)*, which unlike *LINNAPT* and *LINFAPT* does not cater for preserving the patterns. This method is used in our experiments only to evaluate the effectiveness of *LINNAPT* and *LINFAPT*.

RNAT adds noise having zero mean and a uniform distribution. For example, if the domain size of an attribute is $|D|$, *RNAT* generates a pseudorandom 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 (*LINNA* or *LINFA*) is maintained through a wrap around approach when a perturbed value falls outside the domain.

The wrap around approach is explained as follows. If a perturbed value p is greater than the upper limit of the domain $[a, (a + D)]$ the wrap around approach first measures the difference $d = p - (a + D)$. It then produces the final perturbed value $p_f = a + d - 1$. A similar approach is taken if p is less than a. *RNAT* adds noise to all values of an attribute, both to *LINNAs* and *LINFAs* without maintaining the range defined by the conditional values of a *LINFA*.

5.5 Conclusion

In this chapter we have presented a two novel noise addition techniques for non-class numerical attributes. We have carried out some small experiments on the techniques presented in this chapter. Our experimental results are very encouraging. They clearly indicate the effectiveness of the techniques ($LINNAPT$ and $LINFAPT$) in preserving a high data quality in a perturbed data set. However, since we run comprehensive experiments in Chapter 7 on these techniques along with other techniques such as the class attribute perturbation technique, we present experimental results only in Chapter 7. Before we present our main experimental results in Chapter 7 we introduce another technique for noise addition to non-class categorical attributes in Chapter 6.

Chapter 6

Non-class Categorical Attributes Perturbation Technique

6.1 Introduction

So far we have considered data sets having a categorial class attribute and a number of numerical non-class attributes. Therefore, in the previous chapters we have proposed noise addition techniques for such data sets. However, there are many data sets having both numerical and categorical non-class attributes, or only categorical non-class attributes. Examples of such data sets include *Large Soybean Database*, *Banding Database* and *Annealing Data*, which are available from the UCI Machine Learning Repository [77]. We recall that these data sets are required to be released for various purposes such as research, marketing, health care and finance. Ideally, the whole data set should be released without any access restriction so that users can apply various data mining techniques. A disclosure of general patterns and properties of a data set is not a breach of privacy, they are to be extracted from the data set. However, a disclosure of confidential individual values with sufficient certainty is considered a breach of privacy. For the protection of individual privacy in such data sets we propose to perturb all categorical non-class attributes, along with all numerical non-class attributes and the categorical class attribute. However, due to the absence of any natural ordering in categorical values, it is not clear how to add a small amount of noise to them.

The organization of the chapter is as follows. In the next section we present a few existing noise addition techniques (for categorical attributes) including the so-called PRAM. In section 6.3 we discuss a few existing categorical attribute clustering techniques including *CACTUS, ROCK* and *CORE* and we comment on their strengths and limitations. We also discuss the essence of another group of clustering techniques such as *AMOEBA, AUTOCLUST* and *AUTOCLUST+*. In Section 6.4 we present *DETECTIVE*, a novel clustering technique which we use for adding noise to the data set while preserving the patterns. We introduce *EX-DETECTIVE*, a categorical attribute perturbation technique in Section 6.5. Experimental results are presented in Section 6.6. Section 6.8 gives concluding remarks.

6.2 Background

Noise addition for preserving confidentiality of a categorical attribute was first introduced by Warner in 1965 [111] as a technique for protecting individual privacy of the participants in a survey, through noise addition to their responses. We describe this method using the same notation as Bentley [10]. A survey participant is asked two questions, Q and its complement Q^c, where each of them is a *yes/no* question. Examples of Q and Q^c can be *"Are you a smoker?"* and *"Are you a non-smoker?"* respectively. The participant provides an answer to either Q or Q^c. The participant is equipped with a biased coin which has probability p for heads and probability $(1 - p)$ for tails. The probability p is known to the interviewer. The participant tosses the coin (unobserved by the interviewer) and if he/she gets heads he/she answers question Q, otherwise answers question Q^c. As the interviewer does not know which question is answered, the participant is more likely to provide the correct information. It is shown that from such randomized responses of a number of participants a reasonably accurate proportion belonging to *yes* (or *no*) of Q can be estimated - provided participants give correct answers.

In 1977 Kooiman et al. introduced another technique for adding noise to categorical values, the so called Post Randomization Method (PRAM) [61]. PRAM changes the values of a categorical attribute in all records according to a predefined probability distribution. Like many other noise addition techniques PRAM can also result in a learning bias due to the changes of values. We will use the notation of de Wolf et al. [21, 43] in order to describe PRAM. Let ξ denote an attribute in an original data set, let D_ξ denote the domain of ξ and let $K = |D_\xi|$ be the the number of values in D_ξ. Applying PRAM on ξ produces the perturbed attribute X, where the domain of X is the same as the domain of ξ, that is, D_ξ

$= D_X = D$. If the probability that an original value k is changed into a perturbed value l is $p_{kl} = p(X = l \mid \xi = k)$, then $\sum_{l \in D} p_{kl} = 1$ for all $k \in D$. The probability p_{kl} is called the "transition probability". The basic concept of PRAM can be expressed by a $K \times K$ Markov matrix, where the elements are the transition probabilities, as follows.

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ p_{K1} & p_{K2} & \cdots & p_{KK} \end{bmatrix}$$

It is very important to carefully choose the transition probabilities so as to control the amount of noise added to data. In [40] the authors suggest that clustering categorical attribute values can be used to assist in finding an appropriate set of transition probabilities. For example, for some $1 > a > b > 0$, $p_{kl} = a$ if k and l belong to the same cluster and $p_{kl} = b$ if they belong to different clusters.

Giggins and Brankovic [40] proposed a technique to introduce an ordering among the values belonging to a categorical attribute and thereby facilitating noise addition to the values. The main idea is to break down the categorical attribute into sub-attributes that have natural orderings. For example, an attribute *City* with domain {*Sydney*, *Melbourne*, *Brisbane*, *Newcastle*} can be broken down into the following numerical sub-attributes: *Population*, *Geographical Longitude and Latitude*, and *Pollution Index* [11]. Natural orderings of the numerical sub-attributes are then used to assign some ordering among the categorical values. The idea was originally proposed by Brankovic [11] for allowing "range" queries involving categorical attributes.

Both above techniques use the same transition probability for a pair of values belonging to an attribute over all records of a data set. Therefore, clustering is performed only once, involving all records of the data set. However, we observe that two values may not be similar (i.e. may not belong to the same cluster) over the whole data set, but can still be similar within a horizontal segment (set of records) of the data set. For example, *Nurse* and *Secretary*, as two categorical values of an attribute *Profession*, can be considered similar among people older than 60 years - since both of these jobs were typically performed by females and neither of them required tertiary degrees in the past. However, as these jobs are nowadays performed by males and females alike and nursing now requires a Bachelor degree, they may not be similar for the younger part of the population.

In the next section we discuss a few existing categorical attribute clustering techniques including *CACTUS*, *ROCK* and *CORE* and we comment on their strengths and limitations.

6.3 An Overview of Existing Categorical Attribute Clustering Techniques

In this section we discuss a few existing clustering techniques for records with categorical values. We also discuss some limitations of these techniques in the context of noise addition.

We first explain *CACTUS* proposed by Ganti, Gehrke and Ramakrishnan [38]. Let us consider a data set D with three attributes A, B and C having domains $a = \{a_1, a_2, a_3\}$, $b = \{b_1, b_2, b_3, b_4\}$ and $c = \{c_1, c_2\}$, respectively - as shown in Figure 6.1. Note that Figure 6.1 shows the domains of the attributes rather than the data set. *CACTUS* first calculates $E_D(a_1, b_1)$, which is the "expected number" of records having the co-appearance of two values a_1 and b_1 belonging to two different attributes of the data set D. $E_D(a_1, b_1)$ is calculated for the case when the attributes are independent, and all their values are equally likely (the so called Attribute Independence Assumption). Note that *CACTUS* does not require the data set to comply with the Attribute Independence Assumption. On the contrary, in such a data set *CACTUS* would not find any cluster of size greater than one. Therefore, $E_D(a_1, b_1) = \frac{|D|}{|a| \times |b|}$, where $|D|$ is the number of records of the data set. In our example in Figure 6.1, for a data set having 360 records, $E_D(a_1, b_1) = 30$.

If two values $a_i \in a$ and $b_i \in b$ appear together δ times more than $E_D(a_i, b_j)$, where δ is a user defined constant - then the values are considered to be strongly connected. For a δ=1.2, two values a_i and b_j are strongly connected if they co-appear in at least (1.2 $\times$ 30) = 36 records. Let us assume that a_1 and b_1 co-appear in 100 records, a_2 and b_1 co-appear in 31 records, and a_3 and b_1 co-appear in 39 records in the data set D. Therefore, a_1 and b_1, and a_3 and b_1 are strongly connected while a_2 and b_1 are not, as shown in Figure 6.1 by thick and thin solid lines. *CACTUS* regards two values from the same attribute as similar with respect to another attribute, if both values are strongly connected with a common value of the other attribute. For example, a_1 and a_3 are similar with respect to b_1 since both of them are strongly connected with b_1. In Figure 6.1 a similarity between two values has been shown by a dotted line.

CACTUS consists of three phases: *summarization*, *clustering* and *validation*. In *sum-*

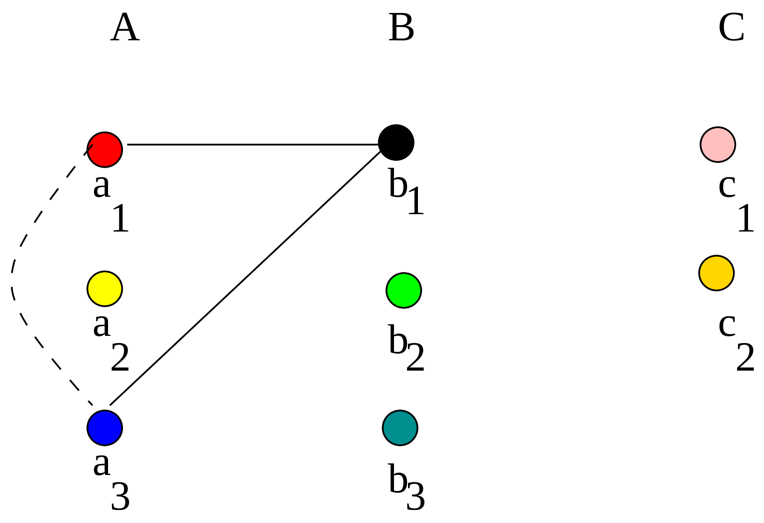

Figure 6.1: The basic concept of similarity, of two values belonging to a categorical attribute, in *CACTUS*.

marization 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. Then, it creates candidate clusters on all the attributes from these individual cluster projections. It initially produces a candidate cluster on a pair of attributes, then extends the pair to a set of three attributes, and so on. 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 given 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. *CACTUS* clusters Cartesian product of attribute domains. It actually considers 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. According to *CACTUS*, an attribute value may belong to a number of clusters. Additionally, a record can also belong to more than one cluster.

We also observe a few characteristics of *CACTUS* which make it unsuitable for usage in noise addition. *CACTUS* considers the whole data set in order to measure if two values are strongly connected. However, we note that two values may not be strongly connected within the whole data set, but they can still be strongly connected in a particular horizontal segment of the data set. *CACTUS* overlooks such strong connections within various segments of a data set. Furthermore, the definition of a cluster used in *CACTUS* is overly restrictive and hence *CACTUS* may end up with huge number of clusters [16]. 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 two of the attributes are approximately equally likely.

Chuang and Chen [16] proposed a correlation based agglomerative hierarchical clustering algorithm called *CORE*. It is a bottom up strategy which initially considers each record as a cluster and then gradually merges these atomic clusters into larger clusters, until a termination condition is satisfied. Chuang and Chen noticed that the correlation between attribute values can be explored to build clusters of records. *CORE* considers a record as a set of attribute values that comprise the record. In Figure 6.2 we illustrate the basic concepts used in *CORE*. The figure shows an example of a data set having four records and three attributes. The dotted lines represent the records and the nodes represent the attribute values of the records.

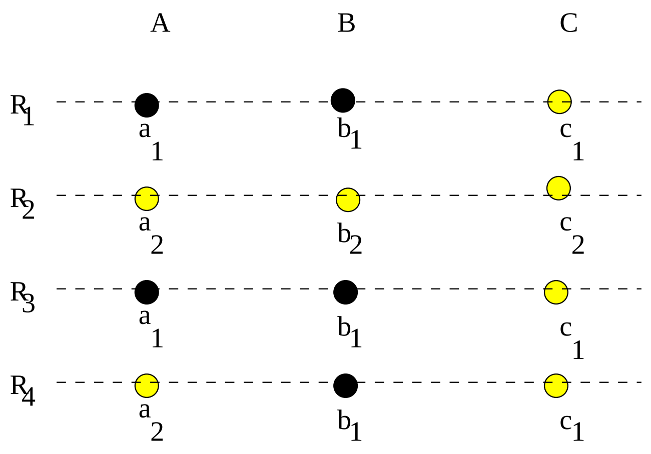

Figure 6.2: An illustration of the correlation analysis by *CORE*.

CORE computes correlations for every pair of attribute values (belonging to different attributes) as a ratio of the number of records having both of the values, to the number of distinct records having any of the values. For instance, in Figure 6.2 the correlation between two attribute values a_1 and b_1, $Corr(a_1, b_1)$ is $\frac{2}{3}$, since they co-occur in two records (R_1 and R_3) and there are three records (R_1, R_3 and R_4) that contain at least one of these two values. For each pair of values a_1 and b_1, *CORE* computes the so called Force Correlation

$FC(a_1, b_1) = Corr(a_1, b_1) - \delta$, where δ is a user defined constant. 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 the Force Correlations for all pairs of values, such that one value of a pair belongs to one cluster and the other value belongs to the other cluster. Two clusters with the largest similarity are merged together. *CORE* then goes into the next hierarchical merging iteration. This process continues until a termination condition is satisfied. Note that each record can only belong to one cluster. However, any value of an attribute may appear in several clusters.

We observe a feature of *CORE* which makes it unsuitable for noise addition. Recall that *CORE* calculates correlation between two values as a ratio of the number of records with both of the values, to the number of records with either of the values. Consider a scenario where one value appears in a very large number of records and the other appears in a small number of records, and when the second value always appears with the first value. In that case the second value implies the first value. *CORE* overlooks such a strong relationship. This is further illustrated in Figure 6.3. According to the figure, the $Corr(a_1, b_1)$ is $\frac{1}{3}$, which is less than their correlation in the previous example shown in Figure 6.2. Since they are having a relatively lower correlation it is possible that they can be considered as repulsive depending on the value of δ. However, considering the fact that a_1 always appears with b_1, a_1 is actually attractive/correlated to b_1. The appearance of a_1 almost certainly indicates the appearance of b_1. Therefore, we argue that *CORE* does not capture this kind of relationship.

Guha, Rastogi and Shim [45] proposed an agglomerative hierarchical clustering algorithm called *ROCK* in the context of Market Basket data set. *ROCK* measures similarity of two records by the ratio of the number of attribute values appearing in both records to the number of distinct attribute values appearing in any of the records. Informally, let s_1 be the number of attributes in which R_1 and R_2 (where R_1 and R_2 are two records) have the same value, and let s be the total number of attributes. Then the similarity, $sim(R_1, R_2)$, of R_1 and R_2 can be expressed as $\frac{s_1}{2s - s_1}$. For example, in Figure 6.3 the $sim(R_1, R_2) = \frac{0}{6}$ and the $sim(R_1, R_3) = \frac{3}{3}$. *ROCK* considers two records as neighbors if their similarity exceeds a given threshold θ. It assigns links between every two neighbor records R_a and R_b, where the number of links $link(R_a, R_b)$ between them is the same as

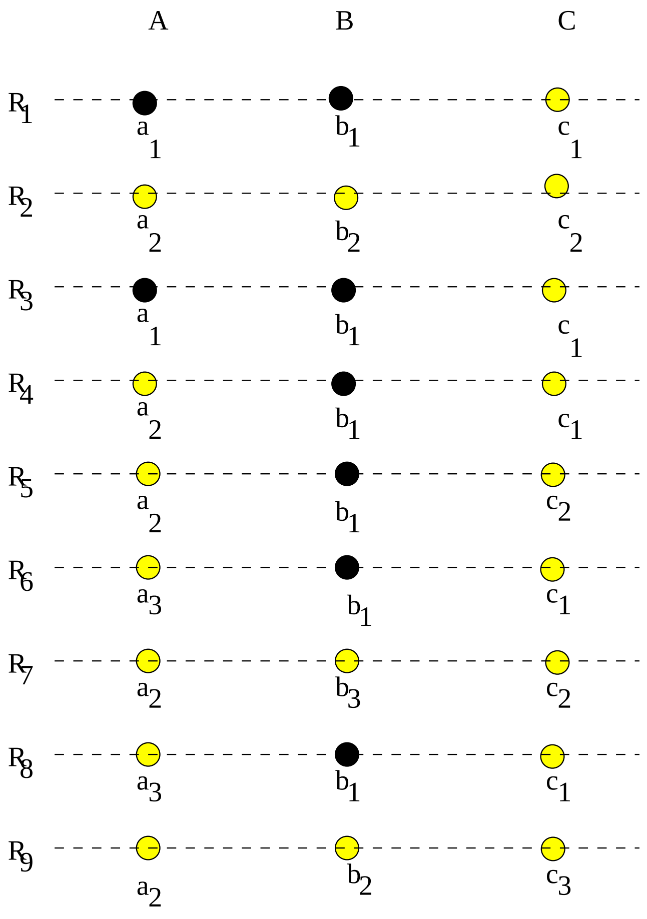

Figure 6.3: An example showing a limitation of *CORE*.

the number of their common neighbors. *ROCK* aims to maximize the sum of links for records belonging to the same cluster, and to minimize the sum of links for records belonging to different clusters. Therefore, the criterion function that *ROCK* wants to maximize is $E_l = \sum_{i=1}^{k} n_i * \sum_{R_a, R_b \in C_i} \frac{link(R_a, R_b)}{n_i^{1+2f(\theta)}}$, where C_i denotes cluster i of size n_i, where $1 \leq i \leq k$.

In the criterion function the total number of links between pairs of records in a cluster is divided by the expected number of these links. This division forces the records having few links between them to be assigned to different clusters. Otherwise all records of a data set could be assigned to a single cluster so as to increase the number of links in the cluster. Based on the criterion function *ROCK* estimates the so-called "goodness measure" for a pair of clusters C_i, and C_j. The goodness measure $g(C_i, C_j)$ of a pair of clusters is the number of cross links between the clusters divided by the expected number of cross links between them that is, $g(C_i, C_j) = \frac{link[C_i, C_j]}{(n_i+n_j)^{1+2f(\theta)} - n_i^{1+2f(\theta)} - n_j^{1+2f(\theta)}}$, where n_i and n_j are the numbers of records in the corresponding clusters, and $link[C_i, C_j]$ is the number of cross links between the clusters. *ROCK* uses the goodness measure for merging the clusters. Note that each record belongs to only one cluster. However, an attribute value may appear in several clusters.

ROCK requires the desired number of clusters k as an input. It then draws a random sample (of size bigger than k) from the data set. 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 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. This process 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. The user defined threshold is difficult to determine without prior knowledge of the data set [16]. We argue that large number of links indicates the existence of high number of common neighbors rather than the similarity between the records. It is possible for a record to be clustered with records which are less similar to it than other records out of the cluster. Additionally, for a low threshold a large number of links between two records may be established without indicating sufficient similarity between them. Besides, *ROCK* assumes that all attributes are categorical and does not take advantage of natural ordering among numerical values.

Dutta, Mahanta and Pujari [27] proposed an agglomerative hierarchical clustering algorithm called *QROCK* - a quick version of the *ROCK* algorithm. The underlying principle of *QROCK* is essentially the same as the underlying principle of *ROCK*. However, the termination condition has been simplified in *QROCK*. In *ROCK* the algorithm terminates when either existing clusters do not have any link/s across them or the user defined number of clusters are obtained. Dutta et al. pointed out that if *ROCK* algorithm terminates by the former criterion, then the final clusters are simply the connected components of the graph that represents the data set. This graph is constructed by considering the records as the vertices; two vertices a and b are connected by an edge if and only if there is at least one link between the records R_a and R_b corresponding to the vertices a and b respectively. Dutta et al. argued that out of the two termination criteria used in *ROCK*, the criterion that terminates the algorithm when there are no links between different clusters is more natural. In fact, it is unrealistic to expect a user to specify appropriate number of clusters for a large data set, especially without any prior knowledge of the data set. A more realistic approach is to let a user choose the similarity threshold for determining neighbors and then the algorithm finds out the clusters that correspond to the connected components of the graph. By adjusting the threshold a user can obtain a suitable set of clusters.

QROCK produces same clusters as *ROCK* terminated under the same termination condition. However, the algorithm of *QROCK* is completely different from the algorithm underlying *ROCK*. *QROCK* simply constructs a graph from the given data set and the user defined threshold θ, and then finds connected components in the graph.

As *QROCK* just needs to compute the neighbor list of each record and does not need to perform several other tasks such as to compute number of links between the records and build local and global heaps - it is a very fast and simple algorithm. The overall complexity of the algorithm is O(n^2), where n is the number of records. On the other hand, the complexity of *ROCK* is $O(n^2 logn + nm_n m_a)$ where m_n is the maximum number of neighbours, and m_a is the average number of neighbours. Thus, in the worst case the complexity of *ROCK* is $O(n^3)$

Dutta et al. compared the performance of *QROCK* algorithm against the *ROCK* algorithm. They used 8124 records of the *Mushroom* data set and 4544 records of the *Landslide* data set. Both data sets were obtained from UCI machine learning repository. The data sets were truncated into 1000, 2000, 3000 and 4000 records to test the programs for different input sizes. Times in seconds to run the programs on *Mushroom* data set and *Landslide* data set are presented in Table 6.1 and Table 6.2 that were originally presented in [27]. These results demonstrate a much better performance of *QROCK* over *ROCK*.

N	*ROCK*	*QROCK*
1000	2.196	0.278
2000	8.397	1.058
4000	41.398	4.06
8124	161.903	18.907

Table 6.1: Time taken by the whole program in seconds on the mushroom data set.

QROCK is simpler and quicker than *ROCK*. We note that under a given threshold if

N	*ROCK*	*QROCK*
1000	0.677	0.171
2000	2.342	0.647
4000	11.208	2.508
4544	14.71	3.242

Table 6.2: Time taken by the whole program in seconds on the landslide data set.

two records are similar to each other then they are always clustered together by *QROCK*. We also note that *QROCK* does not take into account the number of links between two records. However, for different values of θ, *QROCK* may produce different clusters. Selection of an appropriate threshold θ in *QROCK* is essential.

In 2002 Barbara, Couto and Li [9] proposed *COOLCAT*, which clusters records of a data set into k non overlapping clusters, where k is a user defined number. *COOLCAT* is an entropy based heuristic algorithm for clustering records. *COOLCAT* is based on the fact that entropy of a collection of similar records will be lower than the entropy of a collection of dissimilar records. They aimed to minimize the expected entropy of the whole cluster arrangement.

COOLCAT consists of two phases: initialization phase and incremental phase. In the initialization phase it first draws a random sample from the data set. It then selects records, from the sample, that has maximum entropy. These two records are then placed in two separated clusters. *COOLCAT* then selects the third record which maximizes 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. Finally, in the incremental phase each of the remaining records of the data set is assigned to one of the k clusters such that the expected entropy of the whole arrangement is minimized. In addition to *COOLCAT* a number of information theory based categorical value clustering techniques has been proposed in [7, 65, 6, 42].

In 1998 Gibson et al. proposed a categorical value clustering technique, called *STIRR*, based on the methods from non-linear dynamical systems motivated by spectral graph partitioning [39]. *STIRR* represents a data set by a hyper-graph, where the vertices are the values appearing in the data set and the hyper-edges are the records - as illustrated in Figure 6.4, which is very similar to a figure presented in [9]. The data set shown in the

Figure 6.4 has three attributes A, B and C. The attribute A has three values, that appear in the data set, a_1, a_2 and a_3. Similarly, each of the attributes B and C has three values. All these values have been represented by the vertices of the corresponding hyper-graph. The data set has seven records which are represented by the hyper-edges of the graph.

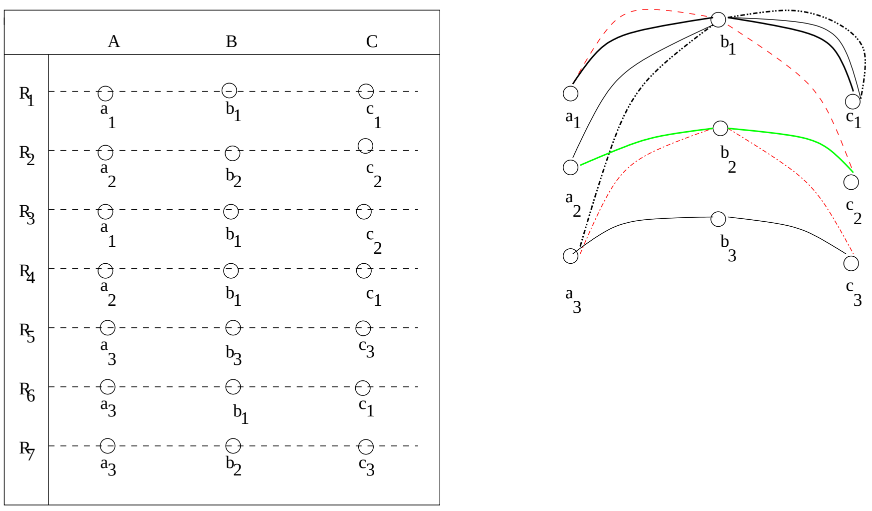

Figure 6.4: Representation of a data set as a hyper-graph.

STIRR performs a clustering of the vertices based on weights. As an end result two values belonging to an attribute are clustered together if they co-appear with the same common values belonging to other attributes.

Many clustering techniques require user defined parameters and prior knowledge on data sets. Such requirements can cause a user created bias, and expensive trial and error steps to find suitable parameter values. Moreover, these requirements contradict the basic concept of exploratory data analysis, where data should “talk” about themselves freely. Estivill-Castro and Lee presented a few clustering techniques such as *AMOEBA* and *AUTOCLUST* that discover clusters automatically without requiring any user defined parameters [29, 30]. Lee and Estivill-Castro [63] also extended their techniques to cluster three dimensional data used in GIS. Additionally, they presented a heuristic to obtain good initial parameter values that are required by their extended technique.

6.4 *DETECTIVE*: A Novel Categorical Values Clustering Technique

In this section we present a novel technique called *DETECTIVE*, A DEcision TreE based CaTegorIcal ValuE clustering technique, which clusters categorical values belonging to an attribute in a given data set. The technique can be applied separately on different categorical attributes. We also present an extension of *DETECTIVE*, called *EX-DETECTIVE*, which allows us to cluster the records of a data set. In many ways, the underlying ideas of *DETECTIVE* and *EX-DETECTIVE* are similar to the underlying ideas of many existing clustering techniques. Nevertheless, there are also significant differences between them as we explain in the following subsections.

6.4.1 The Preliminaries

From the definitions of *Leaf Influential Attributes (LINFAs)* and *Leaf Innocent Attributes (LINNAs)* given in the Introduction of Chapter 5 we remark that records belonging to a leaf have same value in each categorical *LINFA* and the values falling in the same range for each numerical *LINFA*. Among all attributes of the data set *LINFAs* are the relevant attributes as they best describe the class attribute value of the leaf.

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 referred to as the majority class of the leaf. Any other class value that appears in at least one record of a heterogeneous leaf is called a minority class. Furthermore, 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.

6.4.2 DETECTIVE

In order to cluster a categorical attribute, say A, *DETECTIVE* first builds a decision tree (using an existing algorithm such as See5) that considers A as the class attribute. Suppose that it produces a decision tree with l leaves $L_1, L_2, \ldots L_l$. We use R_i to denote a horizontal segment of the data set that is a set of records belonging to L_i.

Suppose L_h is a heterogeneous leaf having the majority class C_p and the minority class C_q. *DETECTIVE* clusters together the values C_p and C_q of class A within the horizontal segment that contains the set R_h. Additionally, let the leaves L_i and L_{i+1} be two sibling leaves (and each of them are homogeneous) having class values C_p and C_r. *DETECTIVE* clusters the class values C_p and C_r within the horizontal segment containing the set of records $R_u = R_i \cup R_{i+1}$. However, unlike traditional clustering techniques, *DETECTIVE* introduces different levels of similarity in different clusters. The class values belonging to a heterogeneous leaf are considered to be more similar than the class values belonging to a pair of sibling leaves, either homogeneous or heterogeneous. Suppose out of two sibling leaves L_i and L_{i+1}, one leaf L_i is heterogeneous having class values C_p and C_q. The class value of L_{i+1} is say, C_r. *DETECTIVE* considers C_p and C_q more similar than C_p and C_r or C_q and C_r.

6.4.3 The Essence

The class values belonging to the records of a heterogeneous leaf are considered similar due to the following reasons. A decision tree building algorithm aims to produce homogeneous leaves. However, in some cases it still produces heterogeneous leaves since any further splitting does not reduce the entropy or the uncertainty about the class values. Entropy of the class values of a set of records depends on the probability distribution of the values in the set. If the distribution is uniform then the entropy is the highest. On the other hand, if all records of the data set have the same value then the entropy is zero. Therefore, the lower the entropy the better the clustering. Since a heterogeneous leaf has the minimum entropy, which can not be significantly reduced by further splitting the leaf, we argue that the class values belonging to the leaf are a cluster within that segment represented by the leaf. We consider these class values so similar that they can not easily be further separated. We remind the reader that here by clustering we mean clustering the values rather than clustering the records. We argue that two values that are similar within one leaf, that is, horizontal segment of the data set may not be similar within another segment. Recall our example where *Nurse* and *Secretary* are two categorical values of an attribute *Profession*. They can be considered similar among people older than 60 years, since both of these jobs were typically performed by females and neither of them required tertiary degrees in the past. However, as these jobs are nowadays performed by males and females alike and nursing now requires a Bachelor degree, they may not be similar for the younger part of the population.

Additionally, each node tests a value of an attribute. If the attribute is categorical then the corresponding node typically has as many children as the domain size of the attribute. This corresponds to dividing the data set, 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 data set is divided into two partitions. Thus, if the node tests a categorical attribute, then the records in each partition contain same values of the attribute, otherwise they contain values that fall within same range. Therefore, the records of a leaf share the same values for all categorical *LINFAs* and the same range for all numerical *LINFAs*. The class values are strongly connected with the value of each of the *LINFAs* (within the segment of the data set) indicating similarity between the class values. We note that essentially the same similarity criterion is used in other categorical clustering algorithm such as *CACTUS* and *STIRR*. What distinguishes our algorithm is the fact that we do not insist on common values in all attributes as a criterion for similarity. In particular, we do not consider leaf innocent attributes. We argue that these attributes do not have significant influence on the class value, as they can not further improve the entropy.

Similarly, the class values belonging to two sibling leaves are also strongly connected with all corresponding values of the *LINFAs* except the one tested in the last node. Hence, class values of the records belonging to two sibling leaves are also considered similar. Moreover, two sibling leaves of a tree can be thought of as a heterogenous leaf obtained by one step pruning of the tree. Therefore, from that point of view, class values belonging to two sibling leaves are also similar. However, since this heterogeneous leaf is artificially created by pruning the tree, this does not represent a natural clustering and therefore indicates a weaker similarity between the values.

The essence of our method is to horizontally partition (cluster) the data set and consider two categorical values of the attribute (that has been considered as the class attribute) similar if they belong to the same partition. We note that in general for different categorical attributes we get different sets of partitions. Our method has two main characteristics: Firstly, similarity is defined within a horizontal partition (attribute specific) only. Secondly, for measuring similarity between categorical values we only consider relevant attributes (the *LINFAs*).

6.4.4 Illustration

We now illustrate the ideas and definitions used in *DETECTIVE* on a few examples (Figure 6.5). We first introduce a synthetically created data set called *Credit Risk (CR)* data set. The *CR* data set has four numerical and one categorical non-class attributes. It also has a categorical class attribute. The numerical attributes are *Income*, *House Rent*, *No. of Dependents*, and *Job Grade*. Their domains are [30, 100], [100, 600], [0, 7], and [1, 4] respectively. 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*}. Detailed properties of *CR* data set have been presented in subsection 6.7.1.

In order to cluster values of the attribute *City*, *DETECTIVE* creates a decision tree that considers *City* as the class attribute. In Figure 6.5 we present a section of the decision tree. There are six internal nodes and five leaves in this section. *Leaf 22* and *Leaf 23* are examples of heterogeneous leaves. There are five records belonging to *Leaf 22* out of which four records have value *Armidale* for the attribute *City*, and the fifth record has a different value. Therefore, majority class of the leaf is *Armidale*. The remaining leaves are homogeneous. *Leaf 19* and *Leaf 20* are siblings. They have exactly the same leaf-path except for leaves themselves and their *LINFAs* include *Credit Risk*, *Income*, and *House Rent*. Similarly, *Leaf 22* and *Leaf 23* are siblings.

Leaf 19 corresponds to a horizontal partition containing four records. Each record has *Sydney* as the value for attribute *City*. *Sydney* always appears with the value *yes* for attribute *Credit Risk* (noted as *risk* in the figure), numerical values falling within the range greater than "eighty two" and less than or equal to "ninety two" for attribute *Income*, numerical values falling within the range less than or equal to "hundred and thirty five" for attribute *House Rent*. However, *Leaf 20* represents another segment of the data set containing 2 records. Each record has *Melbourne* as the value for attribute *City*. This segment is similar to the segment represented by *Leaf 19* except for the attribute *Income*. Therefore, within the union of these two segments *Sydney* and *Melbourne* are similar. However, within the union of two other segments represented by *Leaf 22* and *Leaf 23* (two siblings), *Sydney* and *Armidale* are similar. This illustrates us that different value pairs of attribute City are similar within different segments.

Recall that class values belonging to records of a heterogeneous leaf are considered

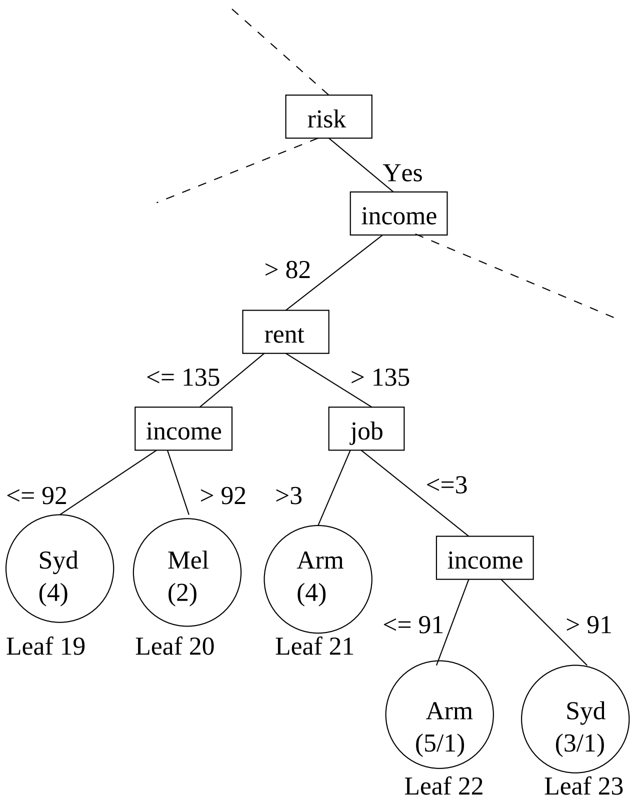

Figure 6.5: A section of the decision tree built on the *CR* data set. The tree considers attribute *City* as class attribute.

similar to each other. Hence, within the segment represented by the heterogeneous leaf *Leaf 22*, *Armidale* is similar to *Sydney*, which is the minority class of the leaf. We consider that the similarity among class values belonging to a heterogeneous leaf is greater than the similarity among class values belonging to two sibling leaves.

DETECTIVE has some basic similarities with existing techniques. It also has some significant differences with them. We discuss the similarities and the differences as follows.

6.4.5 The Similarities

Many existing techniques measure similarity of two values (belonging to a categorical attribute) based on their co-occurrences with a common value of another categorical attribute. Although *DETECTIVE* does not use this concept directly to produce a cluster, two values found similar by *DETECTIVE* co-appear with a common value or common range of values belonging to attributes on the leaf path. In that respect, the underlying idea of *DETECTIVE* is similar to the basic concept of many existing techniques.

Another group of existing techniques rely on the information theoretic approach. In order to cluster the records, these techniques first rearrange them and then horizontally partition the whole data set into a number of clusters in such a way that each cluster upholds a low entropy. Therefore, the entropy of the whole arrangement is minimised. *DETECTIVE* takes a similar approach to clustering categorical values belonging to an attribute by using a decision tree builder which partitions the data set so that in each partition the values belonging to the attribute that we are clustering generate as low entropy as possible. Therefore, *DETECTIVE* ensures that the values, belonging to the attribute, within each horizontal partition are very similar to each other.

6.4.6 The Difference

DETECTIVE differs from existing techniques on few aspects. First, unlike other techniques *DETECTIVE* also takes values belonging to a numerical attribute into account in measuring similarity of two categorical values. If two categorical values co-occur with values (of a numerical attribute) that fall within a same range, then the categorical values are considered similar. Thus, *DETECTIVE* is directly applicable to a data set having both numerical and categorical attributes, unlike most existing techniques which are suitable for data sets having categorical attributes only.

Second, unlike most traditional clustering techniques *DETECTIVE* divides a data set in horizontal partitions, specific to a particular attribute. It extracts similarity of the values belonging to a categorical attribute within each horizontal partition of the data set, instead of within the whole data set. We note that two attribute values may be considered very similar within a particular segment of a data set while they may not be considered similar in the data set as a whole. Third, for measuring similarity between two categorical values *DETECTIVE* focuses on few relevant attributes, the *LINFAs*, instead of all attributes. By "relevant attributes" we mean those attributes which have high influence with the concerned attribute, and which explain the attribute the best within a horizontal segment of the data set.

6.4.7 *EX-DETECTIVE*

DETECTIVE clusters categorical values (rather than records) belonging to an attribute to fulfil its primary objective in supporting noise addition to categorical values. However, traditional clustering techniques usually cluster records and therefore we now present an extended version of *DETECTIVE* called *EX-DETECTIVE* for clustering records.

Just like *DETECTIVE*, *EX-DETECTIVE* first makes use of a decision tree to partition records into leaves for each categorical attribute separately. Then it forms clusters of records such that two records belong to the same cluster if and only if they belong to the same leaf in every such decision tree.

If the data set also has numerical attributes then *EX-DETECTIVE* first computes clusters of records based on all categorical attributes. Within each of these clusters, *EX-DETECTIVE* then applies an existing clustering technique for numerical attributes. However, if the data set does not have any numerical attributes then the clusters obtained from all categorical attributes are used as the final output.

EX-DETECTIVE also allows a data miner to select a set of categorical and numerical attributes to use instead of all attributes of the data set for clustering the records. In that case it first clusters the records based on all selected categorical attributes. Within each cluster it then applies an existing clustering technique on the numerical attributes of the set.

We now explain the steps of *EX-DETECTIVE* on the following example. For simplicity, let us consider a data set having ten records R_1 to R_{10} and three attributes A, B and C, where all of them are categorical. *EX-DETECTIVE* first uses the same decision tree approach as *DETECTIVE* to partition the records based on an attribute A. Let $R_{A1} = \{R_1, R_2, R_3, R_4\}$ (shown in Figure 6.6) be a horizontal segment corresponding to the leaf L_{A1} of the decision tree DT_A that considers attribute A as the class attribute. Similarly, $R_{A2} = \{R_5, R_6, R_7, R_8, R_9, R_{10}\}$ is the segment corresponding to leaf L_{A2}. *EX-DETECTIVE* then uses the same process for another attribute B. Let $R_{B1} = \{R_1, R_2, R_3, R_4, R_5\}$ and $R_{B2} = \{R_6, R_7, R_8, R_9, R_10\}$ be two segments of the data set corresponding to leaves L_{B1} and L_{B2} for decision tree DT_B. *EX-DETECTIVE* produces clusters of records R_{AB1} and R_{AB2} (see Figure 6.6) based on the attributes A and B, where $R_{AB1} = R_{A1} \cap R_{B1} = \{R_1, R_2, R_3, R_4\}$ and $R_{AB2} = R_{A2} \cap R_{B2} = \{R_6, R_7, R_8, R_9, R_{10}\}$. The intersection $R_{A1} \cap R_{B2}$ produces an empty set and the intersec-

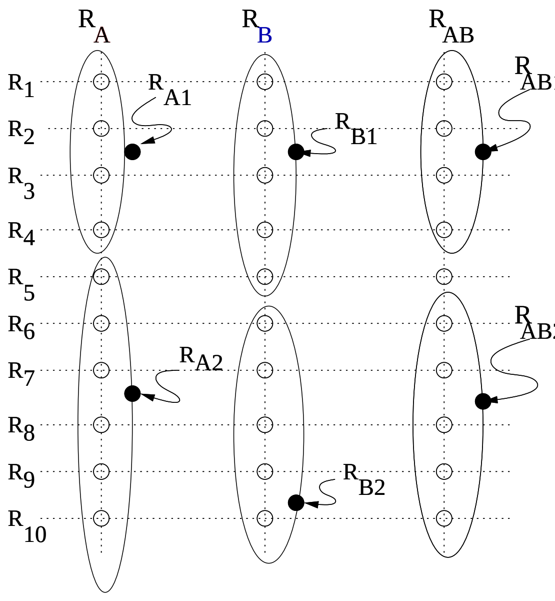

Figure 6.6: Basic steps of *EX-DETECTIVE* - for clustering records based on attributes A and B.

tion $R_{A2} \cap R_{B1}$ produces a set R_5 having one record. These two intersections are therefore not considered as clusters. For determining clusters of records based on all three attributes, *EX-DETECTIVE* again uses *DETECTIVE* to process attribute C. Suppose, the values of attribute C are clustered together in two horizontal segments $R_{C1} = \{R_2, R_3, R_4, R_{10}\}$ and $R_{C2} = \{R_1, R_5, R_6, R_7, R_8, R_9\}$ as shown in Figure 6.7. Therefore, based on all three attributes - we get two final clusters of records $R_{ABC1} = R_{AB1} \cap R_{C1} = \{R_2, R_3, R_4\}$ and $R_{ABC2} = R_{AB2} \cap R_{C2} = \{R_6, R_7, R_8, R_9\}$.

Suppose that along with three categorical attributes the data set also has two numerical attributes D and E. *EX-DETECTIVE* first obtains clusters of records based on all categorical attributes. Within each cluster it then applies a conventional clustering technique such as distance-based clustering and k-means clustering on the numerical attributes only. In Figure 6.8 there are two clusters of the records, R_{ABC1} and R_{ABC2}, based on the categorical attributes. Within each of these segments a conventional clustering technique (for numerical attributes) is applied, and thereby produce the final clusters R_{ABCDE1} and R_{ABCDE2}.

EX-DETECTIVE produces clusters where within each cluster all categorical values

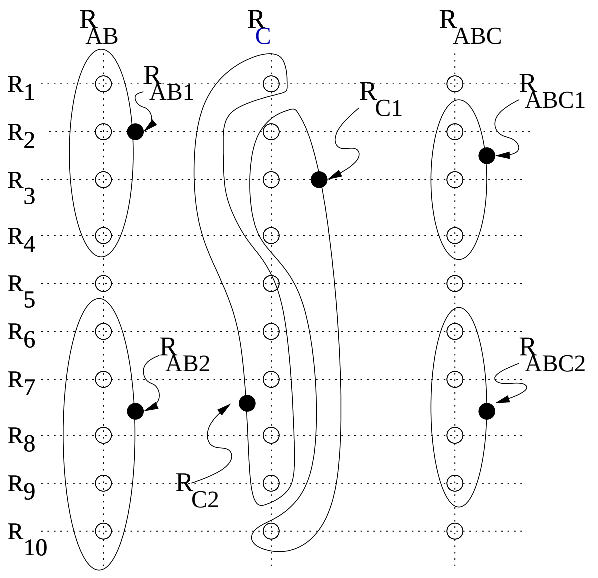

Figure 6.7: Clustering records based on the attributes A, B and C.

belonging to each attribute are similar to each other and all numerical values belonging to each numerical attribute are close to each other. Therefore, clustering records by *EX-DETECTIVE* (based on all attributes) may sometimes be overly restrictive and may produce small number of clusters of small sizes, especially if the data set has too many attributes. Additionally, it is not unlikely that few attributes of a data set will be uncorrelated (non-influential) to other attributes. Presence of such non-influential attributes can cause small clusters. However, a user can assign a weight to each categorical attribute such that the attribute with higher weight has more influence on resulting clusters. In an extreme case, the weight of zero means that the particular attribute will not be taken into account when constructing the clusters. This can be achieved by pruning the tree to the maximum level. The tree with the maximum level pruning has only one leaf that has all records in it and therefore, such a pruning assigns zero weight to the attribute. Less pruning means allocation of more weight and vice versa.

Since *EX-DETECTIVE* can assign various weights to the attributes the final clusterings obtained by it should not be overly restrictive. Moreover, by choosing different sets of weights for the attributes a user can explore different possible clusterings, which in turn can help to extract useful information from a data set. For example, an obvious choice would

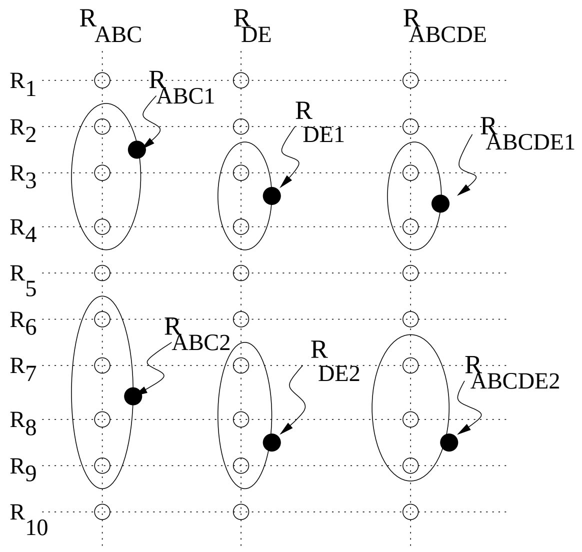

Figure 6.8: Clustering records of a data set having numerical attribute/s along with categorical attribute/s.

be to give more weight to influential attributes than to non-influential ones.

Like many existing techniques, such as *CACTUS*, *EX-DETECTIVE* clusters records using the similarity among the values belonging to an attribute. *EX-DETECTIVE* measures the similarity among the values in a way which is the same as *DETECTIVE.* Therefore, *EX-DETECTIVE* also has the advantages that *DETECTIVE* has. For example, in measuring the similarity of categorical values belonging to an attribute *EX-DETECTIVE* uses few relevant attributes only. It is reasonable to use just the relevant attributes that explain an attribute the best. Additionally, non-influential attributes are still considered by the decision tree builder.

In order to measure the similarities of categorical values belonging to an attribute *EX-DETECTIVE* divides a data set in horizontal partitions, which is specific to the categorical attribute. The significance of such a partitioning is that it allows a user to measure the similarity of categorical values within the partitions separately. Two categorical values may be similar within a horizontal partition while they may not be similar within the whole data set or within another partition. The advantage of the horizontal partitioning has been illustrated in Section 6.2 with the nursing and secretarial job example.

The above mentioned advantages of *EX-DETECTIVE* in measuring similarity of the categorical values belonging to an attribute result in its overall advantages in clustering records, since the clusterings of records are built on the similarities of the values.

6.5 CAPT: Categorical Attributes Perturbation Technique

We now present a technique called *Categorical Attribute Perturbation Technique (CAPT).* It 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 explain *CAPT* with examples as follows. Let us apply *CAPT* on the *CR* data set and perturb the categorical attribute *City*. *CAPT* first uses DETECTIVE on the attribute *City* as the class attribute, and thereby clusters the attribute values. *CAPT* then scans the records of the data set one by one. For each record, it identifies the leaf which the record belongs to. Let us assume that a record belongs to *Leaf 22*, which is a heterogeneous leaf (see Figure 6.5). The majority class of the leaf is *Armidale* while the minority class is *Sydney.* Additionally, *Leaf 22* has a sibling which is *Leaf 23.* The majority class of *Leaf 23* is *Sydney.* *CAPT* either changes the class value of a record belonging to a leaf to the majority class of its sibling leaf or shuffles the class values of the leaf, if the leaf is heterogeneous. *CAPT* changes the class value to the majority class of the sibling leaf with a user defined probability p, and if the leaf is heterogeneous *CAPT* shuffles the class values of the leaf with a probability $(1 - p)$.

For *Leaf 22* *CAPT* changes the class value of a record to *Sydney* with a user defined probability p and it shuffles the class values of *Leaf 22* with a probability $(1 - p)$. While shuffling, regardless of the class value of a record it assigns a value equal to the value of the majority class with a probability $(1 - p) * q$, and assigns a value same as the minority class with a probability $(1-p)*l$. In this case, for the class value of the record it assigns *Armidale* with a probability $(1-p)*q$ while it assigns *Sydney* with a probability $(1-p)*l$. Probabilities q and l are determined by the number of majority records (i.e. number of records having the class values which are same as the majority class) and the number of minority records. Let m be the number of majority records, and n be the number of minority records. *CAPT* calculates q and l as follows: $q = \frac{m}{m+n}$ and $l = \frac{n}{m+n}$.

In case of the existence of more than one minority class in a leaf, *CAPT* assigns the value of the majority class with a probability $(1 - p) * q$, and *ith* minority class with a probability $(1 - p) * l_i$ for the class value of each record belonging to the leaf. Let m be the number of records with majority class, and n_i be the number of records having the ith minority class. *CAPT* calculates the probability q and the probability l_i as follows: $q = \frac{m}{m+k}$, and $l_i = \frac{n_i}{m+k}$, where $k = \sum_{i=1}^{t} n_i$ and t is the number of different minority classes in the heterogeneous leaf.

Let us now illustrate another example where a record belongs to *Leaf 19*, which is a homogeneous leaf having a sibling. In such a case *CAPT* changes the class value (*Sydney*) of the record to the class value (*Melbourne*) of the sibling leaf with a user defined probability p. However, with a probability $(1-p)$ it leaves the class value of the record unchanged. If a leaf has more than one sibling then the above mentioned probability p is equally distributed among the siblings. Alternatively, the probability p could also be distributed among the siblings, proportionally to the number of records in each sibling.

If a heterogeneous leaf does not have any siblings *CAPT* shuffles the class values of the leaf with a probability equal to one. In shuffling class values *CAPT* uses the same approach for any heterogeneous leaf. For any attribute which is totally unrelated to all other attributes, *CAPT* perturbs a value belonging to that attribute to another value with a user defined probability.

We apply *CAPT* on the original data set once for each non-class categorical attribute. Each time it produces a data set with one perturbed attribute in it. From each of these perturbed data sets we take only the values belonging to the perturbed attribute and thereby produce a data set where each non-class categorical attribute is perturbed and all other attributes are unperturbed.

Application of *CAPT* along with the *Leaf Innocent Attribute Perturbation Technique (LINNAPT)*, and *Leaf Influential Attribute Perturbation Technique (LINFAPT)*, and Class Attribute Perturbation Technique (*RPT* or *PPT*) will result in perturbation of all attributes of a data set. Such a perturbation should provide higher security of the data set.

6.6 Experimental Results

We perform two phases of experiments. In the first phase, we apply both *DETECTIVE* and *CACTUS* on a synthetic data set called *CS* data set in order to investigate the quality of *DETECTIVE* in discovering similarity. In the second phase of the experiments, we apply *CAPT* on two synthetic data sets *CR* and *CS* in order to perturb them. For each of these data sets, we then compare the data quality of the perturbed data sets with the data quality of the original data set for estimating the effectiveness of *CAPT* in preserving the data quality while perturbing the data sets. We measure data quality of a perturbed data set by evaluating similarities of the decision trees built on the perturbed data set and on the original data set. We compare a perturbed tree to the original tree by evaluating the similarities of the perturbed rules with the original rules. Based on the degree of similarity of a perturbed rule and the best matching original rule, the perturbed rules are informally categorized in three different types, Type A, Type B, and Type C. We next give an example of a rule set that we shall subsequently use to illustrate Type A, Type B, and Type C.

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

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

- **Type B**: If a perturbed rule differs from its best matching original rule only in conditional values of corresponding numerical attribute/s, then the perturbed rule is defined as Type B rule. For example, a perturbed rule $R_p(2)$: $\mathbf{P}(> \mathbf{6})$ **and** $\mathbf{Q}(\leq \mathbf{2})$ **and** $\mathbf{S}(= \mathbf{s_1}) \Rightarrow \mathbf{c_1}$, differs from its best matching original rule $R_o(1)$ only in the conditional value for attribute **P**. Therefore $R_p(2)$ is defined as Type B.

- **Type C**: If a perturbed rule is significantly different from its best matching original rule then the perturbed rule is defined as Type C. For example, a perturbed rule $R_p(3)$: $\mathbf{P}(> \mathbf{8})$ **and** $\mathbf{Q}(\leq \mathbf{2})$ **and** $\mathbf{R}(< \mathbf{3}) \Rightarrow \mathbf{c_1}$ is significantly different from its best matching original rules $R_o(1)$ and $R_o(3)$ and therefore denote $R_p(3)$ as Type C.

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, a perturbed tree $T_p(1)$ has "Type A" rules which apply to 90% of records and "Type C" rules that apply to the 5% of the records belonging to the perturbed data set. Another perturbed tree $T_p(2)$ has "Type C" 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, we consider that $T_p(1)$ and the original tree are more similar than the $T_p(2)$ and the original tree.

Our experimental results show that the quality of a data set perturbed by *CAPT* remains high. We use See5 decision tree builder to create all decision trees (unless otherwise mentioned) of this section. We now present the details of our experiments with *DETECTIVE* and *CAPT*.

6.6.1 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. Five categorical non-class attributes are *Country of Origin, Car Make, Profession, Parents' Country,* and *Favorite 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, Engineer, Academic}, *Status* ={Good, Bad}. Domain size of each of the attributes *Parents' Country* and *Favorite City* is thirty. These two attributes have randomly distributed values and are not correlated with other attributes. Detailed properties of the *CS* data set have been presented in subsection 6.7.2.

We apply *DETECTIVE* on the *CS* data set in order to cluster values belonging to the attribute *Car Make. DETECTIVE* creates a decision tree, shown in Figure 6.9, that considers *Car Make* as class attribute. 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 the 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

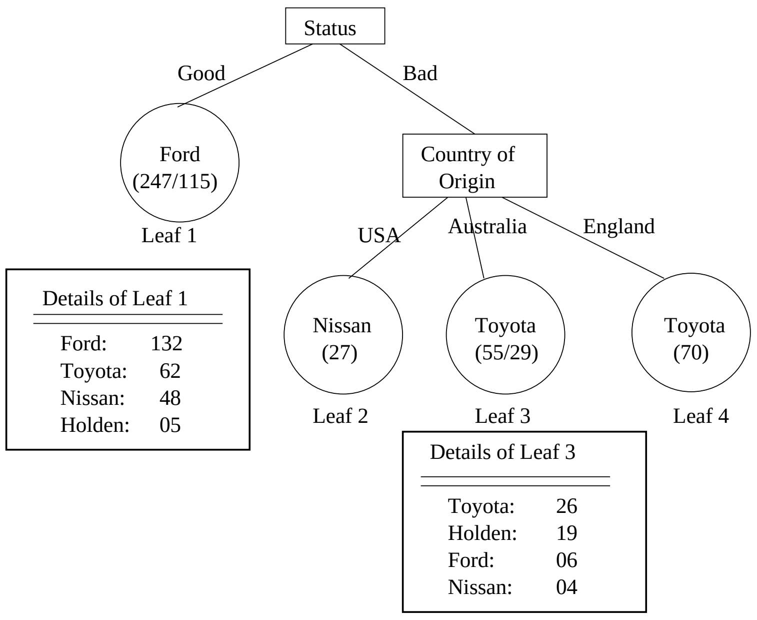

Figure 6.9: Details of a decision tree built from the unperturbed *CS* data set. The tree considers attribute *Car Make* as class attribute. This tree is used for clustering values of the attribute *Car Make*.

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* and *Nissan*.

Country of Origin	Car Make	Profession	Parent's Country	Favorite City	Status
USA	Toyota	Scientist	$\cdots$	$\cdots$	Good
England	Ford	Scientist	$\cdots$	$\cdots$	Good
USA	Ford	Scientist	$\cdots$	$\cdots$	Good
England	Toyota	Scientist	$\cdots$	$\cdots$	Good

Table 6.3: The cluster produced by *CACTUS* from the *CS* data set.

We then apply *CACTUS* on the *CS* data set. We exclude two totally uncorrelated attributes *Parents' Country*, and *Favorite City* from the *CS* data set prior to the application of *CACTUS*, as they would cause large number of small clusters to be discovered. However, for *DETECTIVE* we did not need to exclude such attributes manually. *DETECTIVE* does not use them in clustering, since they do not appear in the decision trees as *Influential Attributes*. While applied on *CS* data set, *CACTUS* produces a cluster shown in Table 6.3. From this cluster we learn that *Toyota* and *Ford* are similar. *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 some sort of measure of similarity level so as to conclude which value pair is more similar than other value pairs.

6.6.2 Experiments on *CAPT*

Experiments using *CR* data set

We use two synthetic data sets; *CR* data set and *CS* data set. We first use 399 records of the *CR* data set. From the original *CR* data set, we create a decision tree T_o that considers the natural class attribute *Credit Risk* as the class attribute. We then apply *CAPT* on the data set and perturb values for the categorical non-class attribute *City*. In all experiments on *CAPT* we use 0.1 as the value of the user defined probability p for changing the class value to the majority class of a sibling leaf/leaves. After perturbation we create another decision tree T_p (from the perturbed data set) that also considers the natural class attribute *Credit Risk* as the class attribute. We then evaluate 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. The differences of the T_ps with the T_o include slightly different number of records belonging to some leaves, slightly different constant values used for numerical attributes in some nodes, and a different order for testing couple of attributes in some 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. Eight (out of ten) perturbed trees have Type A logic rules for more than 50% of the records. Furthermore, eight trees have either Type A or Type B logic rules for more than 90% of the records. Each of the trees has either Type A or Type B logic rules for more than 85% of the records. On the other hand, none of them has Type C logic rules for any of the records. Moreover, the number of misclassified records for any of the perturbed trees is low. As the decision tree algorithms are instable to noise [64], these results suggest the preservation of high data quality in the perturbed data sets.

Experiments using *CS* data set

We then use 399 records of the *CS* data set. From the original *CS* data set we first produce two decision trees, $T_o(car\ make)$ and $T_o(status)$ (shown in Figure 6.9 and in Figure 6.10 respectively), that consider attribute *Car Make* and attribute *Status* as class attribute respectively. We then apply *CAPT* on the *CS* data set 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 data set. We evaluate the similarity of a decision tree built on the perturbed data set with the corresponding decision tree built on the original data set.

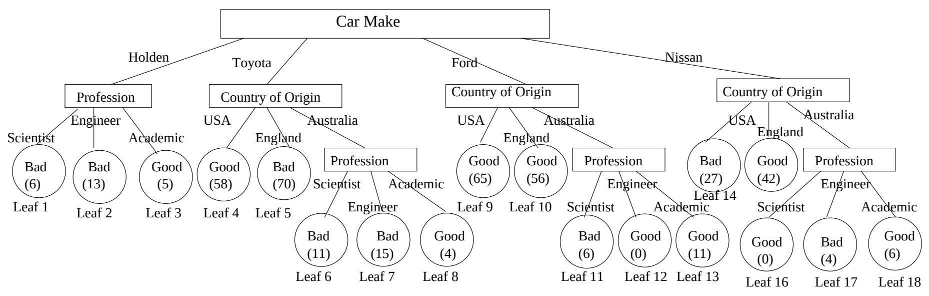

Figure 6.10: A decision tree $T_o(status)$, built on the original *CS* data set. The tree considers the natural class attribute *Status* as class attribute.

We also run this experiment ten times. Each time we produce a decision tree $T_p(car\ make)$ which is extremely similar to the corresponding decision tree $T_o(car\ make)$. All perturbed trees $T_p(car\ make)$s have Type A logic rules covering 100% records. The differences between them (the original and a perturbed tree) include difference in the number of records belonging to a leaf, and number of misclassified records in a leaf.

Moreover, in each of the ten experiments we get a decision tree $T_p(status)$ which is very similar to the decision tree $T_o(status)$. A perturbed tree $T_p(status)$ is shown in Figure 6.11. Logic rules covering almost all records are similar. Occasionally some logic rules, covering smaller number of records, differ partially. Some portions of the perturbed trees are pruned forms of the corresponding portions of the original tree. This suggests loss of some small patterns in the perturbed data sets. However, all big patterns are satisfactorily preserved in all of the perturbed trees. Seven out of ten perturbed trees have Type A logic rules for more than 70% of the records. None of the trees has Type C logic rule for any record. We consider these perturbed trees to be very similar to the original tree.

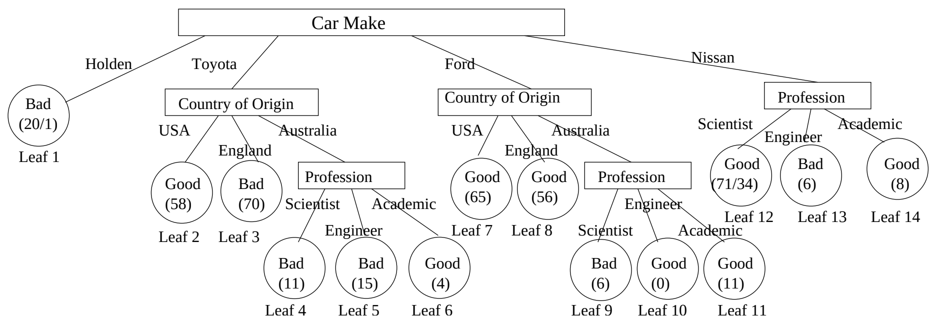

Figure 6.11: A decision tree $T_p(status)$, built on a perturbed *CS* data set. The tree considers the attribute *Status* as class attribute.

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

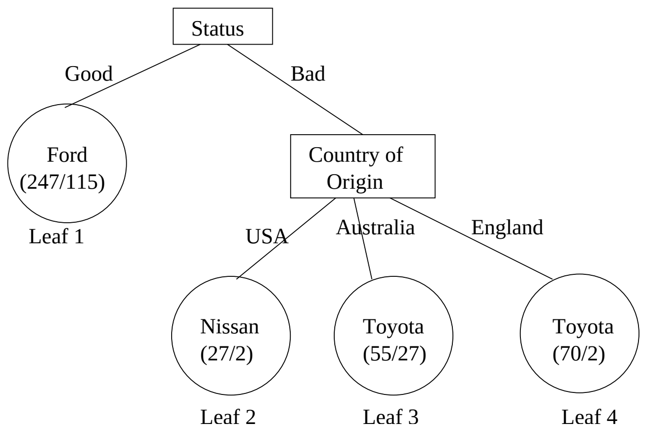

Figure 6.12: A decision tree built on a total perturbed *CS* data set. The tree considers attribute *Car Make* as class attribute.

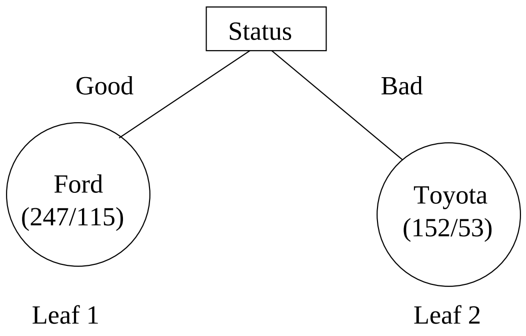

Figure 6.13: A decision tree built on another total perturbed *CS* data set. The tree considers attribute *Car Make* as class attribute.

We run these experiments three times and produce three total perturbed data sets; D_{p1}, D_{p2} and D_{p3}. Out of these three experiments two times we produce decision trees $T_p(car\ make)$s which are identical to the decision tree $T_o(car\ make)$. Moreover, numbers of misclassified records in these perturbed trees are not significantly greater than the number of misclassified records of the original tree $T_o(car\ make)$. One of these perturbed trees $T_p(car\ make)$ is shown in Figure 6.12 while the original tree $T_o(car\ make)$ is shown in Figure 6.9. Out of three experiments once we produced a decision tree $T_p(car\ make)$ which is not extremely similar to the decision tree $T_o(car\ make)$. However, this $T_p(car\ make)$

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

Table 6.4: Similarities of perturbed trees with corresponding original trees.

is just a pruned form of the decision tree $T_o(car\ make)$ where logic rules for most of the records are the same. Logic rules for other records are very similar. The tree is shown in Figure 6.13.

Similarities of other perturbed trees with their corresponding original trees are also very high. We evaluate the similarity of each perturbed tree with the corresponding original tree and present the result in Table 6.4. A perturbed tree has been considered in the table as "Exactly Same" if it has Type A rules for 100% records. If it has Type A rules for at least 60% records and Type C rules for less than 5% records, then the tree has been termed as "Very Similar". However, a perturbed tree has been considered as "Similar" if it has Type A rules for more than 15% records and Type C rules for less than 5% records. Finally, a "Dissimilar" perturbed tree has Type C rules for more than 10% records and Type A rules for less than 10% records.

Experiments using J48 decision tree builder

CAPT uses See5 decision tree builder. From the results presented above we learn that a user employing See5 tree builder can obtain patterns (from a perturbed data set) which are very similar to the original patterns. However, a user could employ any tree builder (instead of See5) on the perturbed/released data set. Therefore, we investigate if a user (applying another tree builder on a perturbed data set) can obtain similar patterns that he/she would obtain from the original data set. In this investigation, we first use a decision tree builder called J48 of WEKA, which is available from The University of Waikato web site [79, 117].

We first build the original decision trees $T_o(status)$, $T_o(car\ make)$, $T_o(country)$ and $T_o(profession)$ from the original *CS* data set using J48. We then use those three total perturbed data sets D_{p1}, D_{p2} and D_{p3}, where attributes *Car Make*, *Country of origin* and *Profession* are perturbed by *CAPT* using See5 decision tree builder. From each total perturbed data set we build a set of trees; $T_p(status)$, $T_p(car\ make)$, $T_p(country)$ and $T_p(profession)$ using J48. Finally, the similarity of each perturbed tree with the corresponding original tree is evaluated and presented in Table 6.5. The result shown in Table 6.5 indicates that a user employing J48 on a perturbed data set is likely to obtain the patterns, which are similar to the patterns he/she would obtain from the original data set using J48.

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

Table 6.5: Similarities of perturbed trees with corresponding original trees - using J48.

6.7 Properties of Synthetic Data Sets

6.7.1 Properties of Credit Risk (CR) Data Set

Each record of the Credit Risk (CR) data set has been created as follows.

```
House Rent = a randomly generated integer value between 100 and 600;

if(House Rent <= 300){
    Income = a randomly generated integer value between 30 and 100;
    if(Income > 50){
        Job Grade= a randomly generated integer value between 1 and 4;
        if(Job Grade <= 2)
          Credit Risk = No;
        else
          Credit Risk = Yes;
        No of Dependents = a randomly generated integer value between 0 and 7;
        City = A value is generated with 25% probability for each of the four
               possible values (Mel, New, Syd, Arm);
    }// end of if(Income > 50)
    if(income <= 50){
        City = A value is generated using the following
               probability distribution.
              33.33% probability for New,
              33.33% probability for Mel,
              33.33% probabiliry for Arm,
              0% probability for Syd.
        if (City == New)
             Credit Risk = No;
        else
             Credit Risk = Yes;
        Job Grade = a randomly generated integer value between 1 and 4;
        No of Dependents = a randomly generated integer value between 0 and 7;
    } // end of if(income <= 50)
} // end of if(House Rent <= 300)

if(House Rent > 300){
    City = A value is generated using the following
           probability distribution.
           50% probability for Mel,
           50% probability for Syd,
           0% probability for New,
           0% probability for Arm;
    if (City = Mel){
        Credit Risk = No;
        No of Dependents = a randomly generated integer value between 0 and 7;
        Income = a randomly generated integer value between 30 and 100;
        Job Grade = a randomly generated integer value between 1 and 4;
    } // end of if(City = Mel)
    else if (City = Syd){
        No of Dependents = a randomly generated integer value between 0 and 7;
        if (No of Dependents <= 2)
          Credit Risk = No;
        else
          Credit Risk = Yes;
        Income = a randomly generated integer value between 30 and 100;
        Job Grade = a randomly generated integer value between 1 and 4;
    }\\ end of else if (city = Syd)
} \\ end of if(House Rent > 300)
```

6.7.2 Properties of Customer Status (CS) Data Set

Each record of the Customer Status (CS) data set has been created as follows.

```
Country of Origin = A value is generated using the following
                    probability distribution:
                    40% probability for USA,
                    40% probability for England,
                    20% probability for Australia;

if (Country of Origin = USA or England){

      Profession = A value is generated using the following
                   probability distribution:
                    95% probability for Scientist,
                    3% probability for Academic,
                    2% probability for Engineer;

      Car Make = A value is generated using the following
                 probability distribution:
                  40% probability for Toyota,
                  40% probability for Ford,
                  20% probability for Nissan;


      if(Country of Origin = USA) {
          if (Car Make = Toyota or Ford)
                Status = Good;
          else
                Status = Bad;
      } // end of if (Country of Origin = USA)

      if(Country of Origin = England){
          if(Car Make = Nissan or Ford)
               Status = Good;
          else
               Status = Bad;
      } // end of if(Country of Origin = England)

} // end of if (Country of Origin = USA or England)

else if (Country of Origin = Australia){

     Profession = A value is generated using following
                  probability distribution:
                   40% probability for Engineer,
                   30% probability for Academic,
                   30% probability for Scientist;

     if (Profession = Engineer){
          Car Make = A value is generated using the following
                     probability distribution.
                     50% probability for Toyota,
                     50% probability for Holden;
          Status = Bad;
     }

     else if (Profession = Scientist or Academic){

          Car Make = A value is generated with 25% probability for each of the four
                     possible values (Toyota, Nissan, Ford, Holden);

          if (Profession = Scientist)
                  Status = Bad;
          else
                  Status = Good;
     }
} // end of else if (Country of Origin = Australia)

Parents' Country = A value is randomly generated from the domain with
                   each value having equal probability;

Favorite City = A value is randomly generated from the domain with
                each value having equal probability;
```

6.8 Conclusion

In this chapter we have presented a categorical attribute values clustering technique called *DETECTIVE*. Another technique for clustering records of a data set having categorical attributes called *EX-DETECTIVE* has also been presented. Finally, a categorical values perturbation technique called *CAPT*, that makes use of *DETECTIVE*, has been presented. All experimental results presented are very encouraging and suggest that *CAPT* perturbs the categorical values for preserving privacy while maintaining the original data patterns to a reasonable level. *CAPT* can be applied along with existing perturbation techniques proposed in the previous chapters in order to perturb all non-class (both categorical and numerical) attributes and class attribute of a data set for higher security.

In the next chapter we present a framework for perturbing all non-class (both numerical and categorical) and class attributes altogether. We also present an extended framework that incorporates *GADP* or *EGADP* in order to preserve the existing correlations, as well, among the numerical attributes.

Chapter 7

The Framework and Experimental Results

Each of the techniques presented so far adds noise either to a class attribute, non-class numerical attribute or non-class categorical attribute. We now present a *Framework* that combines these techniques for adding noise to all the attributes of a data set. The *Framework* adds noise in such a way that the patterns discovered by the decision tree built on an original data set are preserved. Additionally, our *Framework* can be extended so as to preserve the correlation among the attributes as well. This extension makes the *Framework* applicable to a wider range of data sets, both those to be used for classification and those used for statistical analysis. Our experimental results, presented in Section 7.3, indicate that the patterns are very well preserved.

7.1 The Framework

The following is a high level pseudocode of our *Framework*.

For each leaf, DO:

Step 1: Add noise to *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 *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} are produced.

Step 5: Produce the combined perturbed records containing all perturbed attributes from p_{s1}, p_{s2}, p_{s3} and p_{s4}.

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 *LINNAs* are perturbed. If a data set does not have any non-class numerical attributes or if a leaf does not have any *LINNAs* then Step 1 produces a set of records which is the same as the original set of records.

Step 3 adds noise to non-class categorical attributes by *CAPT*. When adding noise to a categorical attribute it produces a decision tree, on the original data set, that considers the categorical attribute as the class attribute. *CAPT* then explores the similarities among the categorical values using the decision tree. These similarities are used for adding noise to the categorical attributes of the records belonging to a leaf.

Finally, Step 5 produces 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. The *Framework* repeats the process for all leaves and thereby perturbs the whole data set.

7.2 The Extended Framework

We now propose an extension of our *Framework* in order to maintain statistical parameters as well. We again build a decision tree on the original data set and add noise to the records belonging to each leaf separately. Our *Extended Framework* has the following steps. Note that steps 1 and 2 in the original framework are replaced by a combined step while the remaining 3 steps are unchanged.

For each leaf, DO:

Step 1: Add noise to *Leaf Influential Attributes (LINFAs)* and *Leaf Innocent Attributes*

(LINNAs) of the original records belonging to the leaf using either *GADP*, or *CGADP* or *EGADP* technique [73, 92, 74], where the domain of each *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

In Step 1 we consider the collection of records that belong to the leaf under consideration to be a data set in its own right. The domains of *LINNAs* remain the same as their domains in the original data set. However, the domains of the *LINFAs* are defined by the conditional values for the leaf. Thus, after the perturbation is completed the values of *LINFAs* will remain within the range defined by the conditional values. Step 2, Step 3 and Step 4 are straightforward and similar to their corresponding steps in the original *Framework*.

The *Extended Framework* effectively divides the original data set into horizontal segments (set of records) defined by the leaves of the decision tree built on the data set. The *GADP* method is then applied to the horizontal segment defined by each leaf separately and thereby a set of perturbed horizontal segments are produced. This results in the preservation of the correlations that exists in the original horizontal segment, in a perturbed horizontal segment. Moreover, the records of a perturbed horizontal segment are free from any biases of types A, B, C or D [73] that are discussed in Chapter 3 of this thesis. Such perturbed horizontal segments cause the whole perturbed data set to have correlations (among the attributes) similar to the one that the original data set has. Moreover, the perturbed data set is also free from the above mentioned biases.

The advantage of adding noise leaf by leaf is in the preservation of all the patterns discovered by the original tree. The trade off that we must bear is in certain decrease of the security, as an intruder will have tight bounds on the original values in *LINFAs*. However, there are no such bounds on *LINNAs*.

7.3 Experiments

We first introduce a few terms that we use throughout this section. The decision tree obtained from an original training data set is referred to as 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 it applies to.

We now describe a few parameter values that we use in our experiments. For noise addition by *LINNAPT* we use a normal distribution for the noise with mean $\mu = 0$ and standard deviation $\sigma = 33.33\%$ of the attribute domain size.

Similarly, for noise addition by *LINFAPT* we use a normal distribution of noise with mean $\mu = 0$ and standard deviation $= 33.33\%$ of the new domain size of the *LINFA*. For both *LINNAPT* and *LINFAPT* we wrap around the perturbed value if it falls outside the domain. The domain of a *LINFA* is determined by the conditional values of the attribute. Additionally, for noise addition to a non-class categorical attribute by *CAPT* we use 0.1 as the user defined probability p for changing the value to one of its similar values.

We carry out experiments on our *Framework* and *Extended Framework* in order to evaluate their effectiveness in maintaining a good data quality in a perturbed data set. We perturb a data set by the *Framework*, the *Extended Framework* and other random noise addition techniques.

We evaluate the data quality of a perturbed data set through a few quality indicators such as the similarity of the decision trees obtained from the perturbed and the original data sets, prediction accuracy of the decision tree obtained from the perturbed data set and the similarity of the correlation matrices of the original and the perturbed data set. We explain these indicators one by one as follows.

Similarity of Decision Trees

We compare a perturbed tree to the original tree by evaluating the similarities of the perturbed rules with the original rules. 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.

In what follows, let **P**, **Q**, **R** and **S** be non-class attributes, and let **C** be the class attribute of a data set. Further suppose that **P**, **Q** and **R** are numerical attributes with domains [1,10]. Let **S** and **C** be 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.
$\mathbf{P}(> \mathbf{5})$ **and** $\mathbf{Q}(\leq \mathbf{2})$ **and** $\mathbf{S}(= \mathbf{s_1}) \Rightarrow \mathbf{c_1}$,
$\mathbf{P}(> \mathbf{5})$ **and** $\mathbf{Q}(\leq \mathbf{2})$ **and** $\mathbf{S}(= \mathbf{s_2}) \Rightarrow \mathbf{c_2}$,
$\mathbf{P}(> \mathbf{5})$ **and** $\mathbf{Q}(\leq \mathbf{2})$ **and** $\mathbf{S}(= \mathbf{s_3}) \Rightarrow \mathbf{c_1}$,
$\mathbf{P}(> \mathbf{5})$ **and** $\mathbf{Q}(> \mathbf{2}) \Rightarrow \mathbf{c_2}$, and
$\mathbf{P}(\leq \mathbf{5}) \Rightarrow \mathbf{c_2}$.

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

- **Type B**: If a perturbed rule differs from its best matching original rule
  1) only in conditional values of corresponding numerical attribute/s, or
  2) where 2 or more rules that differ in only one attribute can be combined to produce a rule that differs from some original rule only in combined values. For example, a perturbed rule $R_p(2)$: $\mathbf{P}(> \mathbf{6})$ **and** $\mathbf{Q}(\leq \mathbf{2})$ **and** $\mathbf{S}(= \mathbf{s_1}) \Rightarrow \mathbf{c_1}$, differs from its best matching original rule $R_o(1)$ only in the conditional value for attribute **P**. Therefore, $R_p(2)$ is defined as Type B. Let us give another example of a Type B rule. Let $R_p(21)$: $\mathbf{P}(> \mathbf{6})$ **and** $\mathbf{Q}(> 4)$ **and** $\mathbf{S}(= \mathbf{s_1}, \mathbf{s_2}) \Rightarrow \mathbf{c_2}$ and $R_p(22)$: $\mathbf{P}(> \mathbf{6})$ **and** $\mathbf{Q}(> 4)$ **and** $\mathbf{S}(= \mathbf{s_3}) \Rightarrow \mathbf{c_2}$ then the combining $R_p(21)$ and $R_p(22)$ we get $\mathbf{P}(> \mathbf{6})$ **and** $\mathbf{Q}(> 4) \Rightarrow \mathbf{c_2}$ which is a Type B rule since it differs 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 $R_p(3)$: $\mathbf{P}(> \mathbf{8})$ **and** $\mathbf{Q}(\leq$

2) and R(< 3) ⇒ $\mathbf{c_1}$ 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, where the weight of a rule is the percentage of total records that follow the rule. We illustrate this similarity measure with an example, where 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. 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, we consider that $T_p(1)$ and the original tree are more similar than the $T_p(2)$ and the original tree.

We classify a perturbed tree in four types; Exactly Same, Very Similar, Similar and Dissimilar. If a perturbed tree has Type A rules for 100% records then we classify the tree as Exactly Same. If it has Type A rules for at least 60% records and Type D rules for less than 5% records, then we consider the tree as Very Similar to the original tree. However, a perturbed tree is considered as Similar if it has Type A rules for more than 15% records and Type D rules for less than 5% records. Finally, we consider that a Dissimilar perturbed tree has Type D rules for more than 10% records and Type A rules for less than 10% records.

Prediction Accuracy of the Decision Tree

We build a classifier from the perturbed tree and apply the classifier on the training and testing data sets separately. We also build a classifier from the original tree and apply the classifier on the training and testing data sets. We then compare their accuracies. Ideally, the accuracy of a perturbed classifier should be as good as the accuracy of the original classifier in order to demonstrate a good data quality of the perturbed data set. If two decision trees, obtained from an original training data set and a perturbed data set, are similar to each other and both trees have similar prediction accuracy on a testing data set then we consider the underlying data sets as similar. The data quality of a perturbed data set is considered to be high when the perturbed data set is similar to the original data set.

Correlation Matrices

We also evaluate the data quality of a perturbed data set through the correlation matrices obtained from the original and the perturbed data sets. If these two correlation matrices are similar then the data quality of the perturbed data set is better in the sense that it can be used for statistical analysis. By similarity of matrices we mean the values of corresponding elements are same or close.

We perform the experiments on two data sets; the *Adult* data set and the *Wisconsin Breast Cancer (WBC)* data set [77]. The experiments and the results are presented in the following subsections.

7.3.1 Experiments on the Adult Data Set

The Adult data set DS^o_{adult} has 32,561 records [77] where each record contains information about a person. There are 15 attributes including one class attribute, that has two categorical values, ">50K" and "<=50K". The non-class attributes are *age, workclass, fnlwgt, education, education-num, marital-status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week,* and *native-country.*

We use the minimum and maximum value that appear in the data set as lower and upper bound of the attribute domain, and we have the following domains.

- *workclass* = {*Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked*}
- *education* = {*Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool*}
- *marital-status* = {*Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse*}
- *occupation* = {*Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces*}
- *relationship* = {*Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried*}
- *race* = {*White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black*}

- *sex* = {*Female, Male*}
- *native-country* = {*United-States, Combodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinidad & Tobago, Peru, Hong, Holand-Netherlands*}.

In DS^{o}_{adult} there are 2,399 records having one or more missing values. We delete these records and produce a data set DS_{adult} having 30,162 records. A decision tree DT_{adult} is built from DS_{adult}, applying See5 decision tree builder. 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}. We build a decision tree $DT_{training}$ (shown in Figure 7.1) from $DS_{training}$. We use See5 decision tree builder with the following parameters; *Pruning CF* = 25% and *Minimum* = 200. *Pruning CF* = 25% is the default value for this option and a value larger than this would result in less pruning. The option *Minimum* = 200 specifies that in order for a node to be split further it must have at least two children with at least 200 recordsin each child.

We maintain the same setting for all decision trees throughout the experiments with Adult data set.

The mean vector for the numerical attributes of the *Adult* data set is [38.44, 189669.70, 10.12, 1102.66, 87.39, 40.88] and the correlation matrix is as follows.

$$\begin{bmatrix} 1.0 & -.075 & .046 & .082 & .060 & .104 \\ -.075 & 1.0 & -.043 & .001 & -.011 & -.024 \\ .046 & -.043 & 1.0 & .126 & .079 & .152 \\ .082 & .001 & .126 & 1.0 & -.032 & .082 \\ .060 & -.011 & .079 & -.032 & 1.0 & .049 \\ .104 & -.024 & .151 & .082 & .049 & 1.0 \end{bmatrix}$$

In both the mean vector and the correlation matrix the order of numerical attributes is the same as the order given above.

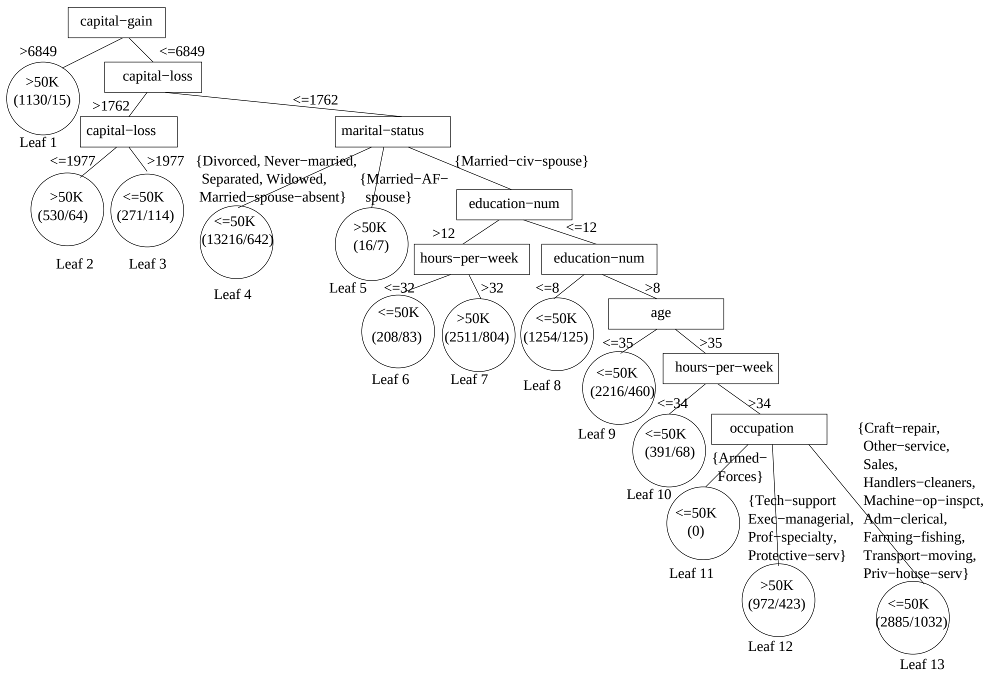

Figure 7.1: The decision tree $DT_{training}$ obtained from 25,600 records of the training *Adult* data set.

Experiments on the Framework

We perturb the *Adult* data set $DS_{training}$ by our *Framework* that incorporates *LINFAPT*, *LINNAPT* (introduced in Chapter 5), *CAPT* (introduced in Chapter 6) and *RPT* (introduced in Chapter 4).

All non-class numerical attributes and the class attribute are perturbed. However, for having a reduced workload we perturb only two categorical attributes *workclass* and *marital-status* by *CAPT*. We run this experiment 5 times and thereby produce five perturbed data sets $DS_{Frmwrk}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{Frmwrk}i$ from a perturbed data set $DS_{Frmwrk}i$, $1 \leq i \leq 5$.

$DT_{Frmwrk}1$ and $DT_{Frmwrk}2$ are presented in the Figure 7.2 and the Figure 7.3. We measure the similarity of the perturbed tree $DT_{Frmwrk}1$ with the original tree $DT_{training}$ by evaluating the types of the rules (revealed in the perturbed tree) and their corresponding weights. For example, $DT_{Frmwrk}1$ has Type A rules that apply to 89.38% of perturbed records and Type B rules that apply to the remaining 10.62% of perturbed records. The tree does not have any "Type D" rules. Similarly, $DT_{Frmwrk}2$ has Type A rules that apply

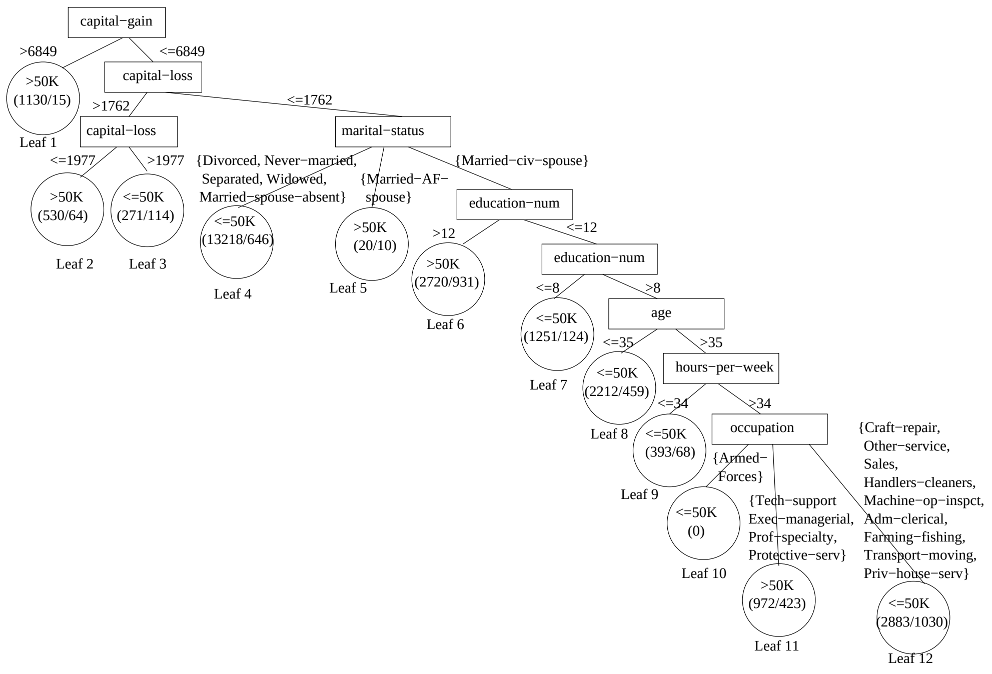

Figure 7.2: The decision tree obtained from a data set perturbed by the *Framework*.

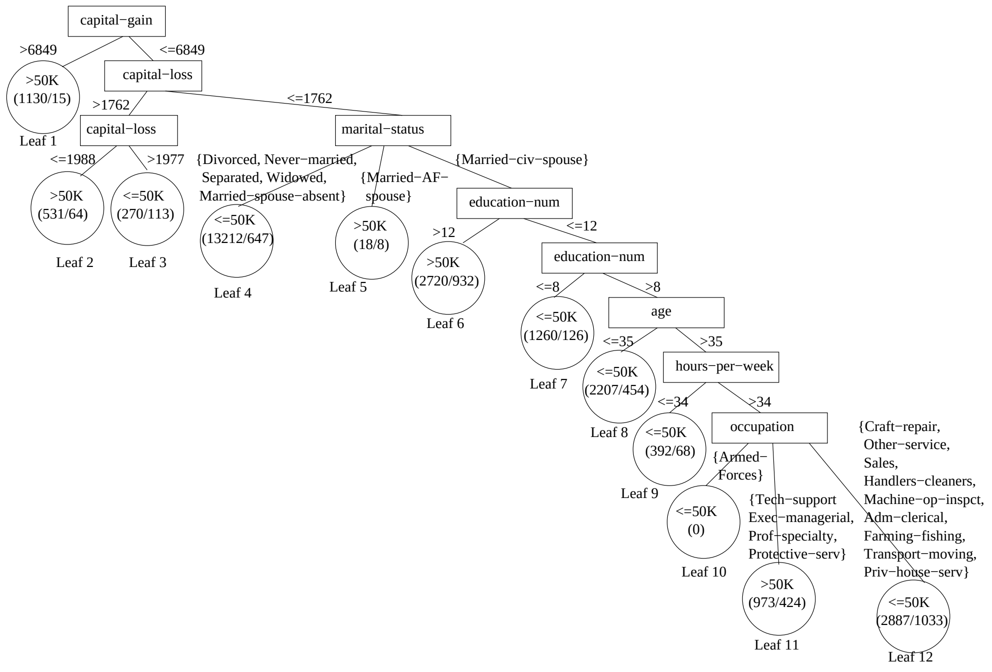

Figure 7.3: The decision tree obtained from a data set perturbed by the *Framework*.

to 86.25% of perturbed records and Type B rules that apply to the remaining 13.75% of perturbed records. $DT_{Frmwrk}2$ also does not have any Type D rules. Therefore, both of these perturbed trees are extremely similar to the original tree $DT_{training}$.

Prediction accuracies of these perturbed trees on the training, perturbed and testing data sets are also very similar to the accuracies of $DT_{training}$. A reader may want to study Table 7.1 for more information. In Table 7.1 names of various classifiers/decision trees are given in the first column. The second, third and fourth columns present the prediction accuracies of a classifier when it is applied on the underlying perturbed data set, original training data set $DS_{training}$, and testing data set $DS_{testing}$, respectively. Since $DT_{Frmwrk}1$ and $DT_{Frmwrk}2$ are very similar to the $DT_{training}$, and the prediction accuracies of $DT_{Frmwrk}1$ and $DT_{Frmwrk}2$ are also very similar to the accuracies of $DT_{training}$ - we consider that $DS_{Frmwrk}1$ and $DS_{Frmwrk}2$ are also very similar to the $DS_{training}$. Therefore, each of these two perturbed data sets maintains a very high data quality.

We study all five perturbed data sets and conclude that all of them maintain very high data quality in terms of similarities of decision trees and their accuracies. In fact, four out of five perturbed trees have either Type A or Type B rules that apply on 100% of the perturbed records. The fifth decision tree has 70% perturbed records having either Type A or Type B rules. None of the five trees have any Type D rules. Prediction accuracies of all these trees are also very similar to $DT_{training}$. For all five perturbed data sets the difference between prediction accuracy of $DT_{training}$ (on $DS_{training}$) and prediction accuracy of $DT_{Frmwrk}i$ (on $DS_{Frmwrk}i$), is less than 0.7% of the total number of records. This difference is less than 0.2% for four out of five trees.

Our *Framework* is tailored for preserving the patterns in a perturbed data set. It does not cater for preserving correlations among numerical attributes. However, we explore the correlations among the numerical attributes of the perturbed data sets. We present a few correlation matrices belonging to the data sets perturbed by the *Framework* as follows.

The mean vector for the numerical attributes of a perturbed data set is [52.13, 716405.71, 9.54, 5644.23, 974.16, 51.81] and the correlation matrix is as follows.

$$\begin{bmatrix} 1.0 & .01 & .001 & .011 & .003 & .03 \\ .01 & 1.0 & -.008 & -.004 & -.004 & -.008 \\ .001 & -.008 & 1.0 & -.017 & -.017 & .069 \\ .011 & -.004 & -.017 & 1.0 & .295 & -.009 \\ .003 & -.004 & -.017 & .295 & 1.0 & -.009 \\ .03 & -.008 & .069 & -.009 & -.009 & 1.0 \end{bmatrix}$$

The mean vector for the numerical attributes of another perturbed data set is [52.31, 717700.64, 9.53, 5551.89, 984.06, 51.82] and the correlation matrix is as follows.

$$\begin{bmatrix} 1.0 & .01 & -.012 & .012 & .009 & .035 \\ .01 & 1.0 & -.007 & .001 & -.001 & .002 \\ -.012 & -.007 & 1.0 & -.009 & -.005 & .077 \\ .012 & .001 & -.009 & 1.0 & .302 & -.007 \\ .009 & -.001 & -.005 & .302 & 1.0 & -.001 \\ .035 & .002 & .077 & -.007 & -.001 & 1.0 \end{bmatrix}$$

The correlation matrices presented here show that the original mean vector and correlations are not preserved in the perturbed data sets. The correlation matrices obtained from other data sets perturbed by the *Framework* are also similar results.

Experiments on a Random Framework

We first introduce a technique called *Random Categorical* that perturbs the categorical attributes. This technique does not cater for preserving patterns. A categorical value is converted to another value belonging to the domain of the attribute, with a user defined probability p. In our experiments we use $p = 0.1$. *Random Categorical* technique does not use the similarities among the categorical value. A value has equal probability for being perturbed as any other values. *Random Categorical* perturbs all values of a categorical attribute with the same probability p.

We combine $RNAT$ (introduced in Chapter 5), *Random Categorical* and $ALPT$ (introduced in Chapter 4) to form a *Random Framework* for perturbing the numerical, categorical and class attributes, respectively. The *Random Framework* is not tailored for preserving pattern of a data set. We use it in order to evaluate the results that we obtain in our experiments on the *Framework*. We perturb only two categorical attributes, *workclass* and *marital status*, in order to make this experiment analogous with our experiments on the *Framework*.

We run this experiment 5 times and thereby produce 5 perturbed data sets $DS_{Random}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{Random}i$ from a perturbed data set $DS_{Random}i$, $1 \leq i \leq 5$. $DT_{Random}1$ and $DT_{Random}2$ are presented in the Figure 7.4 and Figure 7.5, respectively. We again measure the similarity of the perturbed tree $DT_{Random}1$ with the original tree $DT_{training}$ by evaluating the types of rules (revealed in the perturbed tree) and their corresponding weights. $DT_{Random}1$ has Type D rules that apply on 100% records in $DS_{Random}1$. Similarly, $DT_{Random}2$ also has Type D rules that apply on 100% perturbed records. Therefore, $DT_{Random}1$ and $DT_{Random}2$ are very dissimilar to $DT_{training}$.

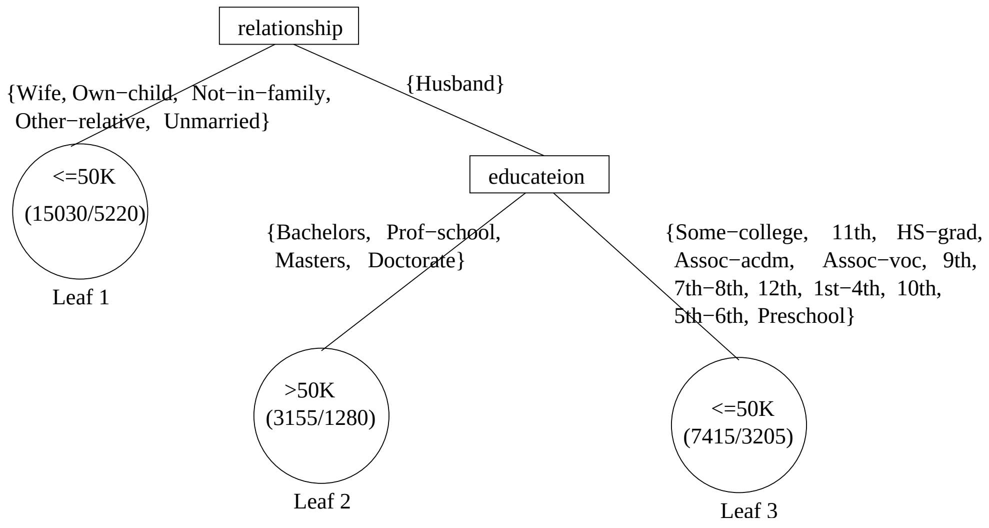
relationship
{Wife, Own−child, Not−in−family, Other−relative, Unmarried}
{Husband}
<=50K
(15030/5220)
Leaf 1
educateion
{Bachelors, Prof−school, Masters, Doctorate}
{Some−college, 11th, HS−grad, Assoc−acdm, Assoc−voc, 9th, 7th−8th, 12th, 1st−4th, 10th, 5th−6th, Preschool}
>50K
(3155/1280)
Leaf 2
<=50K
(7415/3205)
Leaf 3

Figure 7.4: The decision tree obtained from a data set perturbed by the Random Framework.

Prediction accuracies of these trees on the underlying perturbed data sets they have been obtained from are also very poor. See Table 7.1 for more information. Moreover, the prediction accuracies of the trees on the original training data set and the testing data set are also lower than the accuracies of the original tree and the trees obtained from data sets perturbed by the *Framework*. It appears that if all categorical attributes were perturbed the accuracies of the perturbed trees $DT_{Random}1$ and $DT_{Random}2$ could be a lot

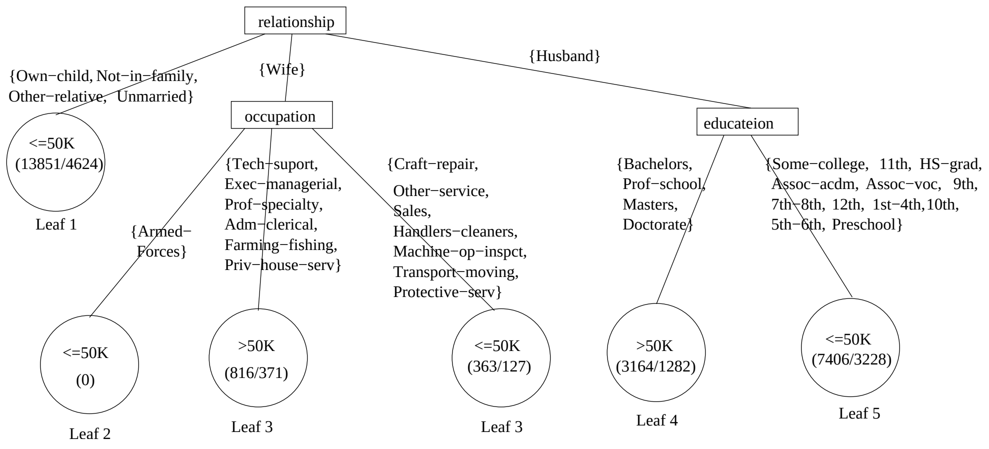
relationship
{Own−child, Not−in−family, Other−relative, Unmarried}
{Wife}
{Husband}
<=50K (13851/4624)
Leaf 1
occupation
educateion
{Armed−Forces}
{Tech−suport, Exec−managerial, Prof−specialty, Adm−clerical, Farming−fishing, Priv−house−serv}
{Craft−repair, Other−service, Sales, Handlers−cleaners, Machine−op−inspct, Transport−moving, Protective−serv}
{Bachelors, Prof−school, Masters, Doctorate}
{Some−college, 11th, HS−grad, Assoc−acdm, Assoc−voc, 9th, 7th−8th, 12th, 1st−4th, 10th, 5th−6th, Preschool}
<=50K (0)
Leaf 2
>50K (816/371)
Leaf 3
<=50K (363/127)
Leaf 3
>50K (3164/1282)
Leaf 4
<=50K (7406/3228)
Leaf 5

Figure 7.5: The decision tree obtained from a data set perturbed by the Random Framework.

worse. The trees mostly use the unperturbed categorical attributes to extract patterns (see Figure 7.4 and Figure 7.5). These may be secondary patterns, which are less prominent than the original patterns extracted by the original tree. These patterns happen to have reasonable prediction accuracies on original training and testing data set. This means that these patterns also exist in the original data sets. However, we note that the prediction accuracies of the perturbed trees on the underlying data sets are very poor. These indicate that the underlying data set is very ”noisy” while at the same time it does not contain strong patterns. As we shall see in the experiments on the next data set that this is not always the case.

We study all five perturbed data sets and their decision trees. We find that all of the decision trees have Type D rules that apply on 100% records of the perturbed data sets. This shows a significant dissimilarity between a perturbed tree and the original tree suggesting a drop in data quality. Additionally, the correlation matrices of the perturbed data sets are also very different from the correlation matrix of the original training data set as expected. The mean vector and correlation matrix of a perturbed data set is shown below.

Mean vector: [55.32, 750087.74, 8.41, 50351.19, 2183.07, 50.06]

Correlation matrix:

$$\begin{bmatrix} 1.0 & 0.0 & -.004 & 0.0 & .010 & -.004 \\ 0.0 & 1.0 & .002 & -.001 & .004 & 0.0 \\ -.004 & .002 & 1.0 & -.007 & -.003 & .001 \\ 0.0 & -.001 & -.007 & 1.0 & -.006 & -.014 \\ .011 & .004 & -.003 & -.006 & 1.0 & .002 \\ -.005 & 0.0 & .002 & -.014 & .002 & 1.0 \end{bmatrix}$$

Experiments on the Extended Framework

We use *GADP* in our experiments on the *Extended Framework*. We apply the *Extended Framework* on $DS_{training}$ and produce a perturbed data set. We run this experiments 5 times and produce 5 perturbed data sets $DS_{ExtndFrm}i$, $1 \le i \le 5$. We build the decision tree $DT_{ExtndFrm}i$ from a perturbed data set $DS_{ExtndFrm}i$.

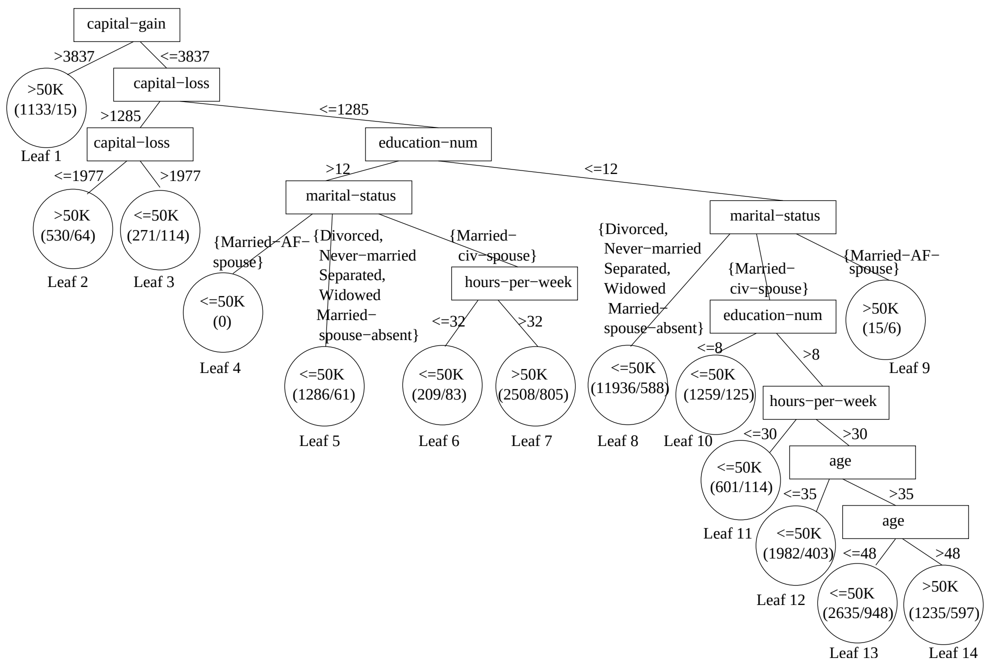

Figure 7.6: The decision tree obtained from a data set perturbed by the *Extended Framework*.

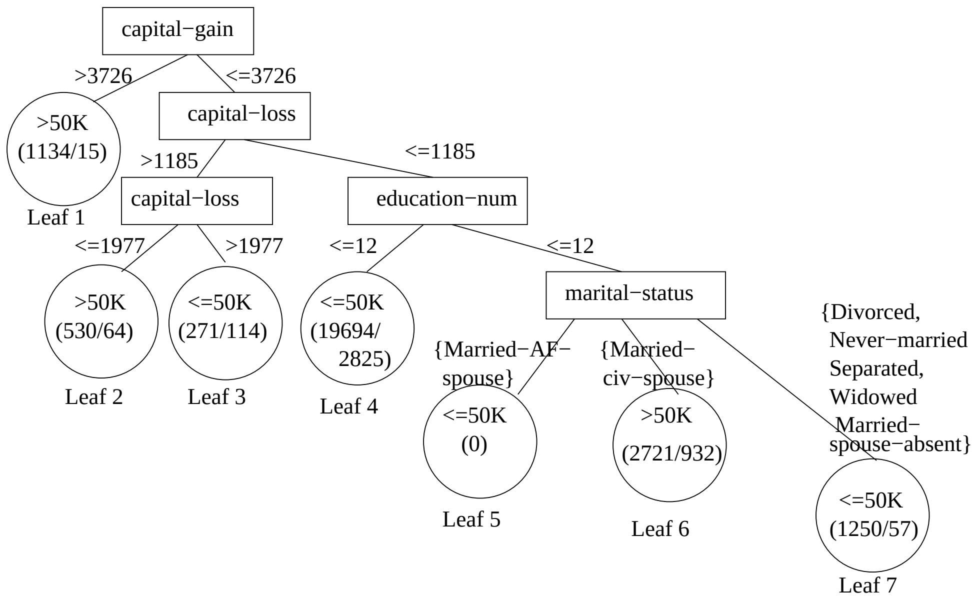

Figure 7.7: The decision tree obtained from a data set perturbed by the *Extended Framework*.

$DT_{ExtndFrm}1$ and $DT_{ExtndFrm}2$ are presented in the Figure 7.6 and Figure 7.7, respectively. $DT_{ExtndFrm}1$ has Type B rules for the records belonging to Leaf 1, through to Leaf 9 of the perturbed tree. Combined logic rules for Leaf 4 and Leaf 9 correspond to the logic rule of Leaf 5 of $DT_{training}$. Thus they fall in the category Type B (see Figure 7.1 and Figure 7.6). Similarly the combined logic rules for Leaf 5 and Leaf 8 jointly correspond to the logic rule for Leaf 4 of $DT_{training}$ and also fall in the category Type B. Therefore, all together $DT_{ExtndFrm}1$ has Type B logic rules that apply on 74.79% of records belonging to $DS_{ExtndFrm}1$. Other perturbed logic rules are of Type C. Therefore, $DT_{ExtndFrm}1$ is very similar to $DT_{training}$.

$DT_{ExtndFrm}2$ has Type B rules that apply on 18.20% of perturbed records. Other rules are also similar to the original rules and thus belong to Type C. For example, the logic rule for Leaf 4 of $DT_{ExtndFrm}2$ (having 77% of the total records) corresponds to the original combined rules for the leaves from Leaf 8 to Leaf 13. The logic rule for Leaf 4 of $DT_{ExtndFrm}2$ belongs to Type C. Moreover, there is no Type D rules in $DT_{ExtndFrm}2$. Therefore, $DT_{ExtndFrm}2$ is also similar to $DT_{training}$. Just like $DT_{ExtndFrm}1$ and $DT_{ExtndFrm}2$ other perturbed trees ($DT_{ExtndFrm}3$, $DT_{ExtndFrm}4$ and $DT_{ExtndFrm}5$) are also similar to $DT_{training}$.

The similarity between $DT_{training}$ and the tree obtained from a data set perturbed by *Extended Framework* is much higher than the similarity between $DT_{training}$ and the tree obtained from a data set perturbed by *Random Framework*. However, similarity between $DT_{training}$ and the tree obtained from a data set perturbed by *Extended Framework* is lower than the similarity between $DT_{training}$ and the tree obtained from a data set perturbed by *Framework*. Therefore, we remark that although the *Extended Framework* produces perturbed data sets having high quality, the quality is not as high as the quality of the data sets produced by the *Framework*.

The prediction accuracies of $DT_{ExtndFrm}1$ and $DT_{ExtndFrm}2$ are higher than $DT_{Random}1$ and $DT_{Random}2$. However, they are slightly lower than the accuracies of $DT_{Frmwork}1$ and $DT_{Frmwrk}2$. See Table 7.1 for more information. For all five perturbed data sets the difference between prediction accuracy of $DT_{training}$ (on $DS_{training}$) and prediction accuracy of $DT_{ExtndFrm}i$ (on $DS_{ExtndFrm}i$), is less than 0.3% of the total number of records.

The *Extended Framework* is tailored for preserving patterns and correlations among numerical attributes. To show that we consider the correlation matrices for the numerical attributes of the perturbed data sets as follows, for each of the five perturbed data sets.

1) For $DS_{ExtndFrm}1$:
Mean vector: [32.76, 195740.46, 9.64, 1538.99, 146.81, 40.08]
Correlation matrix:

$$\begin{bmatrix} 1.0 & -.082 & .079 & .137 & .109 & .165 \\ -.082 & 1.0 & -.043 & .026 & .009 & -.005 \\ .079 & -.043 & 1.0 & .148 & .060 & .144 \\ .137 & .026 & .048 & 1.0 & -.075 & .142 \\ .109 & .009 & .060 & -.075 & 1.0 & .039 \\ .165 & -.005 & .144 & -.142 & .039 & 1.0 \end{bmatrix}$$

2) For $DS_{ExtndFrm}2$:
Mean vector: [38.72, 194131.30, 9.70, 1456.10, 148.43, 40.64]
Correlation matrix:

$$\begin{bmatrix} 1.0 & -.059 & -.002 & .016 & .202 & .085 \\ -.060 & 1.0 & -.046 & 0 & -.008 & .004 \\ -.002 & -.046 & 1.0 & .125 & .062 & .160 \\ .016 & 0 & .125 & 1.0 & -.076 & .088 \\ .202 & -.008 & .062 & -.076 & 1.0 & .032 \\ .085 & .004 & .160 & .088 & .032 & 1.0 \end{bmatrix}$$

3) For $DS_{ExtndFrm}3$:

Mean vector: [35.23, 192430.78, 9.72, 1580.73, 145.91, 40.51]

Correlation matrix:

$$\begin{bmatrix} 1.0 & -.074 & .155 & .034 & .003 & .105 \\ -.074 & 1.0 & -.071 & 0 & -.045 & -.018 \\ .154 & -.071 & 1.0 & .124 & .068 & .134 \\ .034 & 0 & .124 & 1.0 & -.007 & .096 \\ .003 & -.045 & .068 & -.007 & 1.0 & .029 \\ .105 & -.018 & .134 & .096 & .029 & 1.0 \end{bmatrix}$$

4) For $DS_{ExtndFrm}4$:

Mean vector: [43.73, 186317.97, 9.75, 1553.85, 146.33, 41.25]

Correlation matrix:

$$\begin{bmatrix} 1.0 & -.061 & -.020 & .052 & -.133 & -.042 \\ -.061 & 1.0 & -.031 & .013 & .002 & -.016 \\ -.020 & -.031 & 1.0 & .150 & .038 & .165 \\ .052 & .013 & .150 & 1.0 & -.078 & .070 \\ -.133 & .002 & .038 & -.078 & 1.0 & .042 \\ -.042 & -.016 & .165 & .070 & .042 & 1.0 \end{bmatrix}$$

5) For $DS_{ExtndFrm}5$:

Mean vector: [36.84, 195808.26, 9.70, 1517.38, 145.99, 40.46]

Correlation matrix:

$$\begin{bmatrix} 1.0 & -.070 & .043 & .245 & .174 & .186 \\ -.070 & 1.0 & -.072 & -.011 & -.011 & -.023 \\ .043 & -.072 & 1.0 & .128 & .051 & .155 \\ .245 & -.011 & .128 & 1.0 & -.076 & .077 \\ .174 & -.011 & .051 & -.076 & 1.0 & .065 \\ .186 & -.023 & .155 & .077 & .065 & 1.0 \end{bmatrix}$$

For convenience of the reader we repeat the mean vector and correlation matrix from the original training set $DT_{training}$ as follows.

Mean vector: [38.44, 189669.70, 10.12, 1102.66, 87.39, 40.88]

Correlation Matrix:

$$\begin{bmatrix} 1.0 & -.075 & .046 & .082 & .060 & .104 \\ -.075 & 1.0 & -.043 & .001 & -.011 & -.024 \\ .046 & -.043 & 1.0 & .126 & .079 & .152 \\ .082 & .001 & .126 & 1.0 & -.032 & .082 \\ .060 & -.011 & .079 & -.032 & 1.0 & .049 \\ .104 & -.024 & .151 & .082 & .049 & 1.0 \end{bmatrix}$$

We compare the above correlation matrices with the correlation matrix obtained from $DT_{training}$. These experimental results show that the original correlations are not completely preserved in the data sets perturbed by the *Extended Framework*. However, they are significantly better preserved than in the data sets perturbed by the *Framework*. We invite the reader to verify this by inspecting the matrices.

We note that the *GADP* technique may not preserve the original correlations very well if the data set does not have a multivariate normal distribution [92]. Moreover, for a small size data set the technique may not also be very suitable [74]. Recall that *CGADP* or *EGADP* are improved are improved version of *GADP* that are suitable for non-multivariate and small size data sets, respectively. The *Adult* data set does not have a multivariate normal distribution. Besides, the number of records belonging to a leaf is very small. Therefore, *CGADP* or *EGADP* can be used instead.

Experiments on Random Extended Framework (REF)

We introduce *Random Extended Framework (REF)* and compare it with our *Extended Framework*. *REF* applies *GADP* on all records at a time - rather than on the records belonging to a leaf separately. This technique does not preserve the range of an attribute for the records belonging to a leaf.

Random Extended Framework also incorporates *Random Categorical* technique that was used in *Random Framework*. We perturb two categorical attributes; the *workclass* and the *marital-status*. *REF* also makes use of *ALPT* technique to perturb the class attribute.

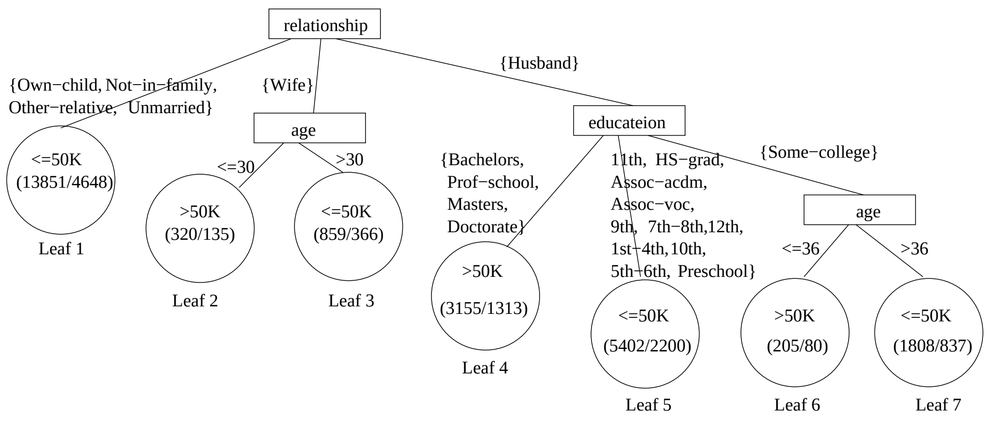

Figure 7.8: The decision tree obtained from a data set perturbed by Random Extended Framework.

We apply *REF* on $DS_{training}$ and produce a perturbed data set. We run this experiment 5 times and produce 5 perturbed data sets $DS_{RandomExt}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{RandomExt}i$ from a perturbed data set $DS_{RandomExt}i$. $DT_{RandomExt}1$ and $DT_{RandomExt}2$ are presented in Figure 7.8 and Figure 7.9. None of these perturbed trees has Type A or Type B rules. All of them have Type D rules that apply on 100% perturbed records. Therefore, the perturbed trees are significantly different from $DT_{training}$ suggesting a noteworthy quality drop in the perturbed data sets.

Prediction accuracies of the perturbed trees are also worse than the accuracies of the trees obtained from data sets perturbed by *Extended Framework* or *Framework*. See Table 7.1 for further information. The perturbed trees also have very high inaccuracies on

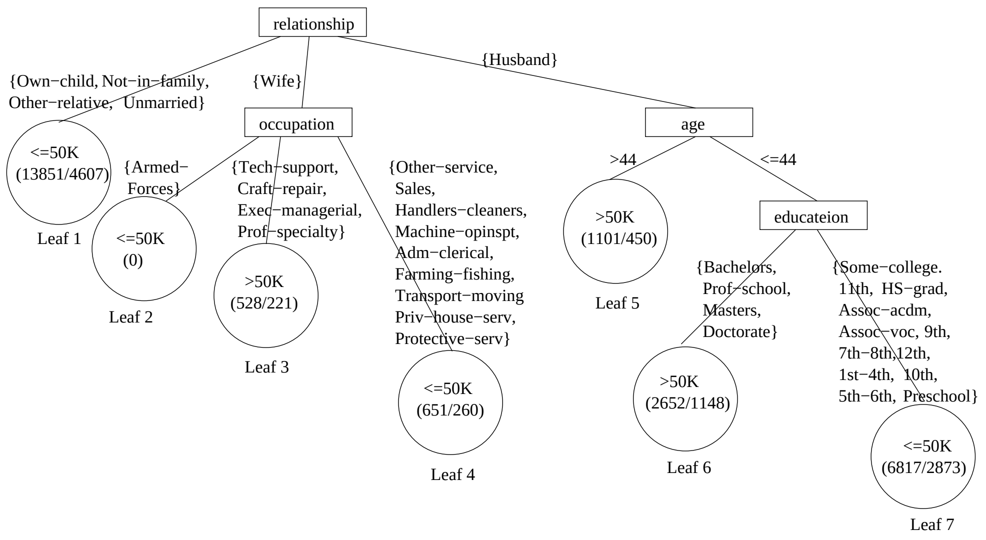
relationship
{Own−child, Not−in−family, Other−relative, Unmarried}
{Wife}
{Husband}
<=50K (13851/4607)
Leaf 1
occupation
{Armed−Forces}
<=50K (0)
Leaf 2
{Tech−support, Craft−repair, Exec−managerial, Prof−specialty}
>50K (528/221)
Leaf 3
{Other−service, Sales, Handlers−cleaners, Machine−opinspt, Adm−clerical, Farming−fishing, Transport−moving Priv−house−serv, Protective−serv}
<=50K (651/260)
Leaf 4
age
>44
>50K (1101/450)
Leaf 5
<=44
educateion
{Bachelors, Prof−school, Masters, Doctorate}
>50K (2652/1148)
Leaf 6
{Some−college. 11th, HS−grad, Assoc−acdm, Assoc−voc, 9th, 7th−8th,12th, 1st−4th, 10th, 5th−6th, Preschool}
<=50K (6817/2873)
Leaf 7

Figure 7.9: The decision tree obtained from a data set perturbed by Random Extended Framework.

the underlying data sets. For all five perturbed data sets the difference between prediction accuracy of $DT_{training}$ (on $DS_{training}$) and prediction accuracy of $DT_{RandomExt}i$ (on $DS_{RandomExt}i$) is less than 22.43% of the total number of records. Recall that the same difference for $DT_{ExtndFrm}i$ is less than 0.3% of the total number of records.

Random Extended Framework is supposed to preserve the correlations among numerical attributes. We next inspect the mean vectors and correlation matrices for a couple of perturbed data sets as follows.

1) For $DS_{RandomExt}1$:

Mean vector: [35.82, 193373.30, 9.60, 3406.13, 205.45, 40.05]

Correlation matrix:

$$\begin{bmatrix} 1.0 & -.074 & .054 & .098 & .094 & .096 \\ -.074 & 1.0 & -.041 & -.012 & -.028 & -.009 \\ .054 & -.041 & 1.0 & .108 & .045 & .147 \\ .098 & -.012 & .109 & 1.0 & -.040 & .051 \\ .094 & -.028 & .045 & -.040 & 1.0 & .038 \\ .096 & -.009 & .147 & .051 & .038 & 1.0 \end{bmatrix}$$

2) For $DS_{RandomExt}2$:

Mean vector: [38.80, 190019.02, 9.63, 3505.15, 206.79, 40.36]
Correlation matrix:

$$\begin{bmatrix} 1.0 & -.063 & .026 & .015 & -.011 & .082 \\ -.063 & 1.0 & -.047 & -.016 & 0 & -.049 \\ .026 & -.047 & 1.0 & .112 & .066 & .148 \\ .015 & -.016 & .112 & 1.0 & -.024 & .077 \\ -.011 & 0 & .066 & -.024 & 1.0 & .046 \\ .082 & -.049 & .148 & .077 & .046 & 1.0 \end{bmatrix}$$

Correlation matrices for other perturbed data sets are also similar to those presented here. It appears that the correlations are better preserved in these data sets than in the data sets perturbed by any other method, which is not surprising considering that *GADP* here operates on the whole data set rather than each leaf separately as in *Extended Framework*.

7.3.2 Experiments on Wisconsin Breast Cancer Data Set

We now run the experiments on *Wisconsin Breast Cancer (WBC)* data set, which is available from [77]. The data set has all together 11 attributes where the class attribute is categorical. The domain of the class attribute is {2,4} where 2 stands for benign and 4 represents malignant. Out of the 10 non-class attributes the 1st one is *Sample Code Number*, which is the id number of each record and therefore it is removed before a release of the data set. The other non-class attributes are *Clump Thickness, Uniformity of Cell Size, Uniformity of Cell Shape, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli* and *Mitoses.* All these non-class attributes in *WBC* data set are numerical and each of them has a domain [1,10]. There are 699 records out of which 16 records have missing values. We delete these 16 records and thus produce a data set DS_{WBC} having 683 records. A decision tree DT_{WBC} is built from DS_{WBC}, applying See5 decision tree builder. We then divide DS_{WBC} into two data sets; a training data set $DS_{training}$ having 600 records and a testing data set $DS_{testing}$ having 83 records. $DS_{testing}$ is created by taking approximately 12% of records belonging to each leaf of DT_{WBC}. The remaining records of DT_{WBC} are used to form $DS_{training}$. We build a decision tree $DT_{training}$ (shown in Figure 7.10) from $DS_{training}$. We use See5 decision tree builder in its default setting for all experiments with *WBC* data set.

The mean vector for the numerical attributes of $DT_{training}$ is [4.44, 3.18, 3.24, 2.88, 3.23, 3.6, 3.41, 2.89, 1.63] and the correlation matrix is as follows.

$$\begin{bmatrix} 1.0 & .63 & .64 & .49 & .52 & .61 & .56 & .53 & .36 \\ .63 & 1.0 & .90 & .72 & .76 & .70 & .77 & .72 & .47 \\ .64 & .90 & 1.00 & .70 & .73 & .73 & .75 & .71 & .45 \\ .49 & .72 & .70 & 1.00 & .59 & .68 & .69 & .62 & .42 \\ .52 & .76 & .73 & .59 & 1.00 & .59 & .62 & .65 & .49 \\ .61 & .70 & .73 & .68 & .59 & 1.00 & .70 & .60 & .34 \\ .56 & .77 & .75 & .69 & .62 & .70 & 1.00 & .68 & .35 \\ .53 & .72 & .71 & .62 & .65 & .60 & .68 & 1.00 & .46 \\ .36 & .47 & .45 & .42 & .49 & .34 & .35 & .46 & 1.00 \end{bmatrix}$$

Experiments on *Framework*

We perturb $DS_{training}$ by our *Framework*. Since *WBC* data set does not have any non-class categorical attribute our *Framework* incorporates *LINFAPT*, *LINNAPT*, and *RPT*. We perturb all numerical attributes and the class attribute. We run this experiment 5 times and thereby produce five perturbed data sets $DS_{Frmwrk}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{Frmwrk}i$ from a perturbed data set $DS_{Frmwrk}i$, $1 \leq i \leq 5$.

$DT_{Frmwrk}1$ and $DT_{Frmwrk}2$ are presented in Figure 7.11 and Figure 7.12. We measure the similarity of the perturbed tree $DT_{Frmwrk}1$ with the original tree $DT_{training}$ by evaluating the types of the rules (revealed in the perturbed tree) and their corresponding weights. For example, $DT_{Frmwrk}1$ has Type A rules that apply to 95.5% of total number of perturbed records and Type D rules that apply to 2.67% of perturbed records. Similarly, $DT_{Frmwrk}2$ has Type A rules that apply to 92.83% of total number of perturbed records

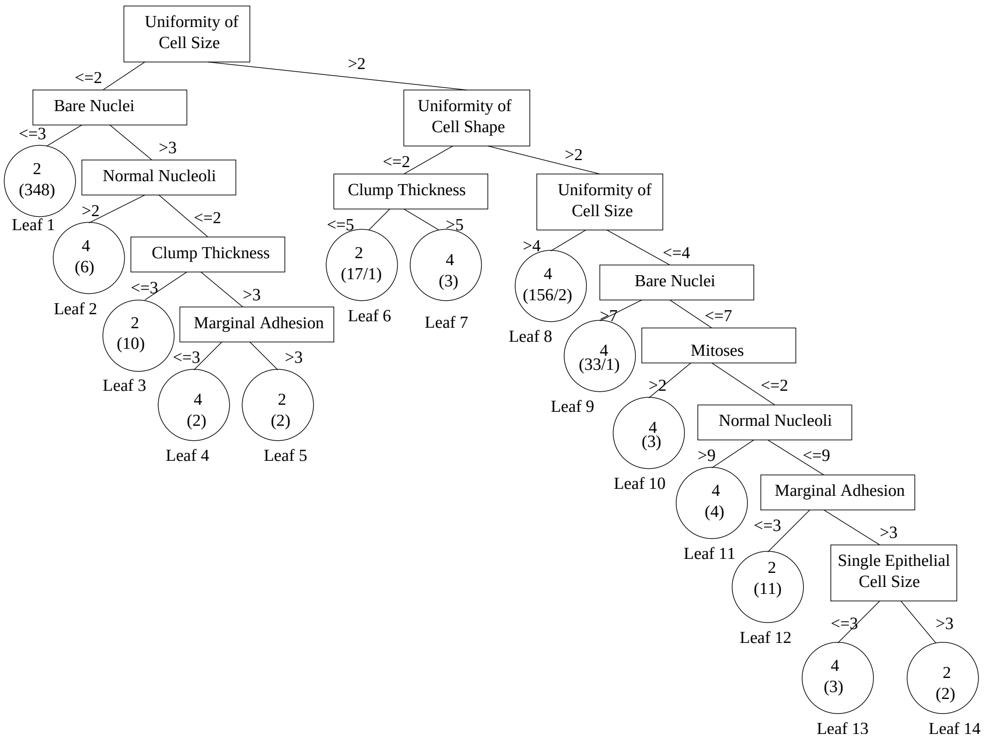

Figure 7.10: The decision tree obtained from the training *WBC* data set.

and Type D rules that apply to 0% of perturbed records. Therefore, both of these perturbed trees are extremely similar to the original tree $DT_{training}$.

Prediction accuracies of these perturbed trees on the training, perturbed and testing data sets are also similar to the accuracies of $DT_{training}$. A reader may want to study Table 7.2 for more information. The second, third and fourth columns present the prediction accuracies of a classifier when it is applied on the underlying perturbed data set on the original training data set $DS_{training}$, and on the testing data set $DS_{testing}$, respectively. Since $DT_{Frmwrk}1$ and $DT_{Frmwrk}2$ are very similar to the $DT_{training}$, and the prediction accuracies of $DT_{Frmwrk}1$ and $DT_{Frmwrk}2$ are also very similar to the accuracies of $DT_{training}$ - we consider that $DS_{Frmwrk}1$ and $DS_{Frmwrk}2$ are also very similar to the $DS_{training}$. Therefore, each of these two perturbed data sets maintains a very high data quality.

We study all five perturbed data sets and conclude that all of them maintain very high data qualities in terms of similarities of decision trees and their prediction accuracies. All of the five perturbed trees have Type A rules that apply on more than 90% of the total number of perturbed records. Four out of the five trees do not have a single record that follows a

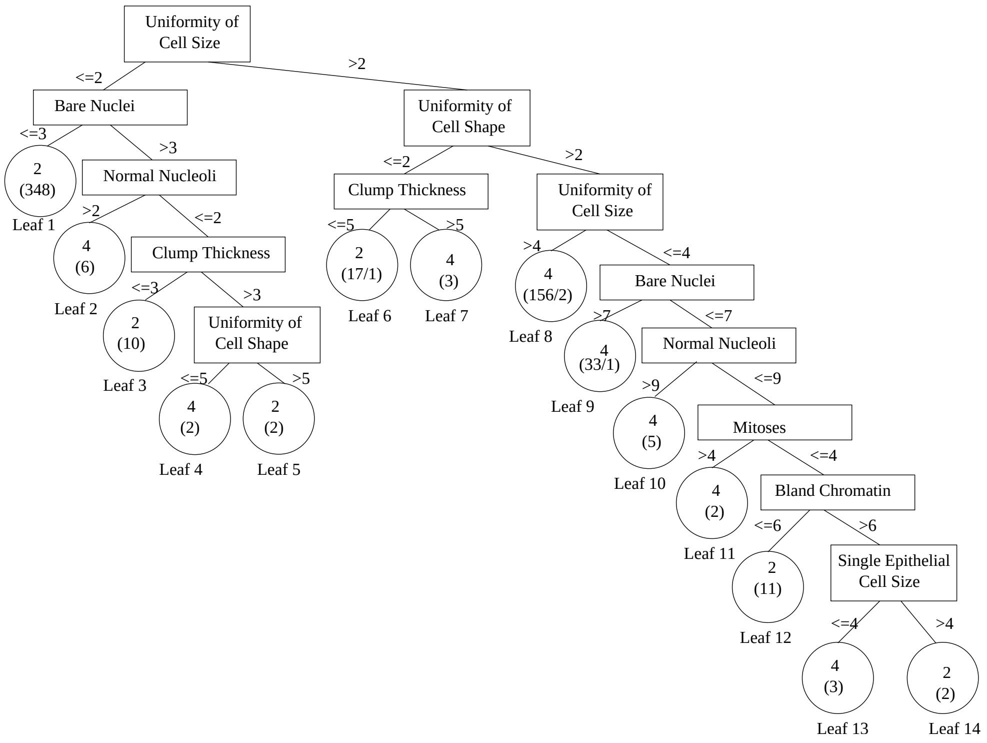

Figure 7.11: The decision tree obtained from a *WBC* data set perturbed by the *Framework*.

Type D rule. The fifth tree Type D rules that apply on only 2.67% of the total number of records. Prediction accuracies of all these trees are also very similar to the accuracy of $DT_{training}$. For all five perturbed data sets the difference between prediction accuracy of $DT_{training}$ (on $DS_{training}$) and prediction accuracy of $DT_{Frmwrk}i$ (on $DS_{Frmwrk}i$), is less than 0.85% of the total number of records.

Our *Framework* is specifically designed to preserve the patterns in a perturbed data set. It does not cater for preserving correlations among numerical attributes. A correlation matrix belonging to a data set perturbed by the *Framework* are presented as follows.

The mean vector for the numerical attributes of $DT_{Frmwrk}1$ is [4.87, 3.26, 4.68, 4.75, 4.44, 3.42, 4.63, 4.63, 4.06] and the correlation matrix is as follows.

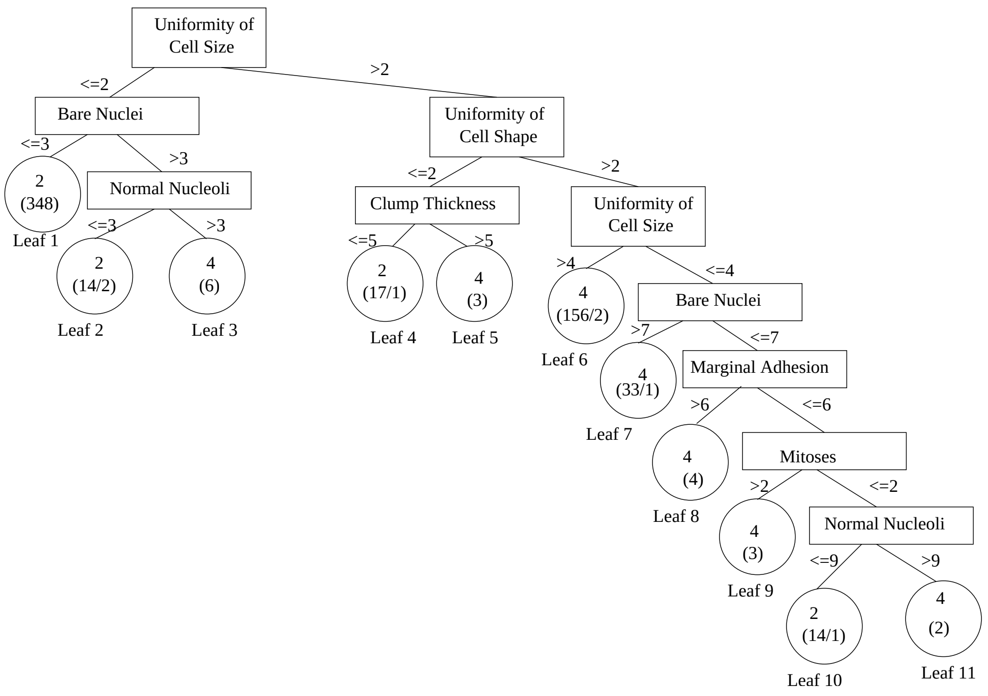

Figure 7.12: The decision tree obtained from a *WBC* data set perturbed by the *Framework*.

$$\begin{bmatrix}
1.0 & .31 & .17 & .15 & .08 & .27 & .16 & .06 & .07 \\
.31 & 1.0 & .36 & .21 & .22 & .59 & .26 & .17 & .14 \\
.17 & .36 & 1.00 & .11 & .09 & .33 & .10 & .05 & -.02 \\
.15 & .21 & .11 & 1.00 & .04 & .16 & .08 & .03 & .00 \\
.08 & .22 & .09 & .04 & 1.00 & .13 & .07 & .06 & .02 \\
.27 & .59 & .33 & .16 & .13 & 1.00 & .25 & .08 & .01 \\
.16 & .26 & .10 & .08 & .07 & .25 & 1.00 & .02 & .07 \\
.06 & .17 & .05 & .03 & .06 & .08 & .02 & 1.00 & .05 \\
.07 & .14 & -.02 & .00 & .02 & .01 & .07 & .05 & 1.00
\end{bmatrix}$$

The correlation matrix presented here shows that the original correlations are not preserved in the perturbed data set, as expected. The correlation matrices obtained from other data sets perturbed by the *Framework* are similar to the one presented here.

Experiments on Random Framework

Since *WBC* does not have any non-class categorical attributes we combine *RNAT* and *ALPT* to form a *Random Framework* for perturbing the attributes of a data set. The *Random Framework* is not tailored for preserving patterns of a data set. We use it in order to evaluate the results that we obtain in our experiments on the *Framework*.

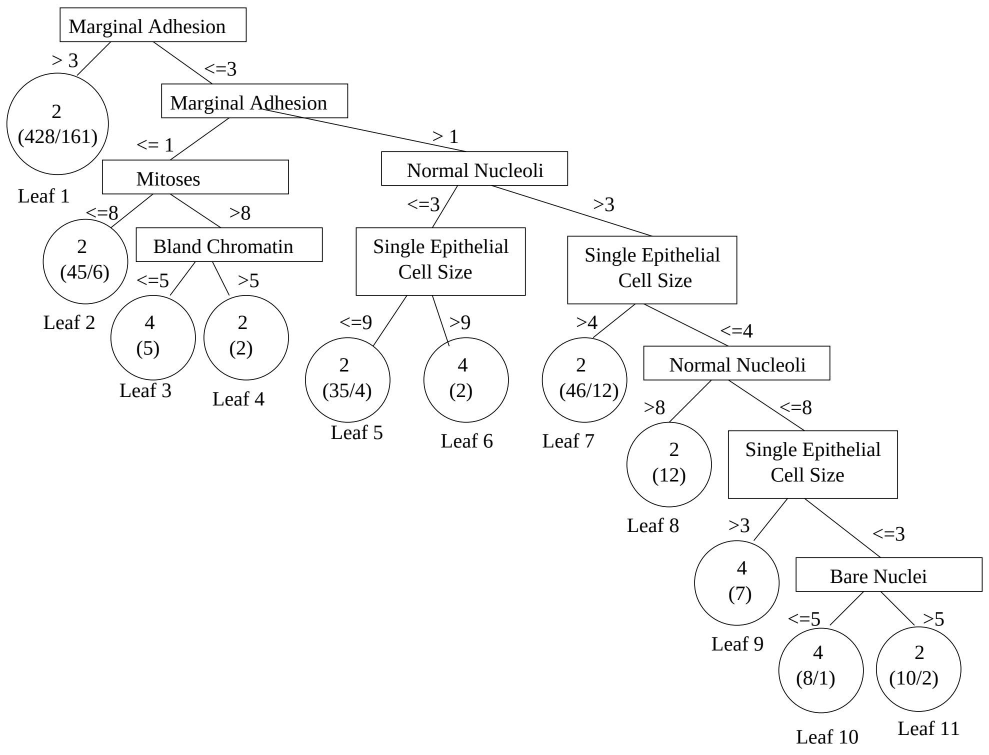

Figure 7.13: The decision tree obtained from a data set perturbed by Random Technique.

We run this experiment 5 times and thereby produce 5 perturbed data sets $DS_{Random}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{Random}i$ from a perturbed data set $DS_{Random}i$, $1 \leq i \leq 5$. $DT_{Random}1$ and $DT_{Random}2$ are presented in Figure 7.13 and Figure 7.14, respectively. We again measure the similarity of the perturbed tree $DT_{Random}1$ with the original tree $DT_{training}$ by evaluating the types of rules (revealed in the perturbed tree) and their corresponding weights.

$DT_{Random}1$ has Type C rules that apply on 100% records in $DS_{Random}1$. $DT_{Random}1$ does not have any Type A or Type B rules that apply on any record of $DS_{Random}1$. $DT_{Random}2$ has Type D rules that apply on 100% perturbed records. Therefore, $DT_{Random}1$

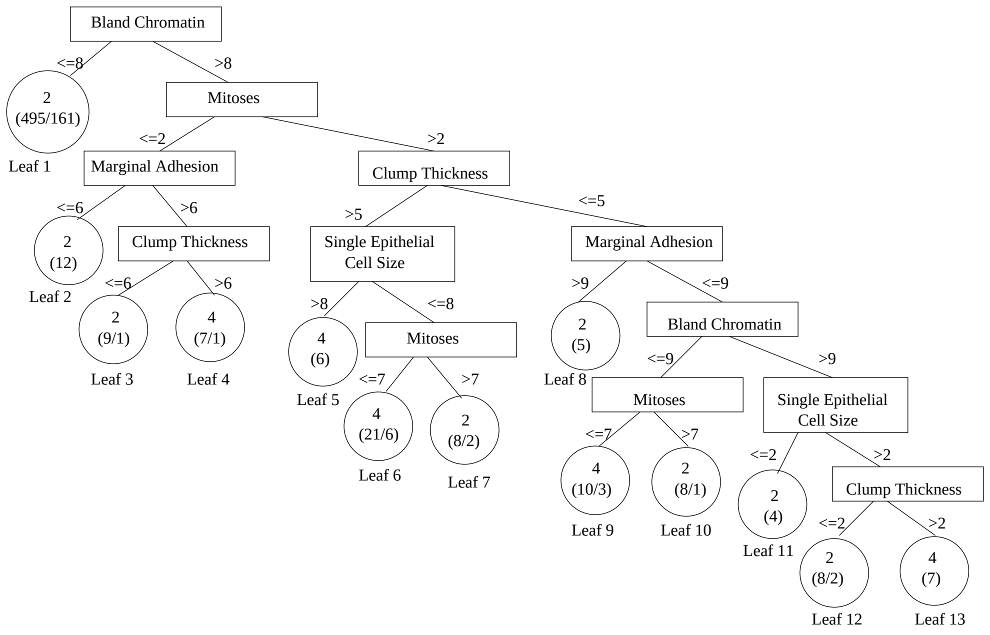

Figure 7.14: The decision tree obtained from another data set perturbed by Random Technique.

and $DT_{Random}2$ are considered very dissimilar to $DT_{training}$.

Prediction accuracy of these trees on the underlying perturbed data sets are also very poor. See Table 7.2 for more information. Moreover, the prediction accuracy of the trees on the original training data set and the testing data set are also very poor. They are significantly worse than the accuracies of the trees obtained from data sets perturbed by *Framework*.

We study all five perturbed data sets and the trees obtained from them. We find that all of the decision trees have Type C and/or Type D rules that apply on 100% records of the perturbed data sets. Two out of five trees have Type D rules that apply on more than 80% of the records. One of these five trees has only a single leaf and therefore, does not extract any pattern. There are altogether two trees having three or less number of leaves. Additionally, there is a tree having 22 number of leaves. Above study indicates a significant dissimilarity between the perturbed trees and the original tree suggesting a huge drop in data quality. Additionally, the correlation matrices of the perturbed data sets are also very different from the correlation matrix of the original training data set. Correlations among the attributes are completely lost in the perturbed data sets.

Experiments on the *Extended Framework*

We use *GADP* in our experiments on the *Extended Framework*. We apply *Extended Framework* on $DS_{training}$ and produce a perturbed data set. We run this experiments 5 times and produce 5 perturbed data sets $DS_{ExtndFrm}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{ExtndFrm}i$ from a perturbed data set $DS_{ExtndFrm}i$.

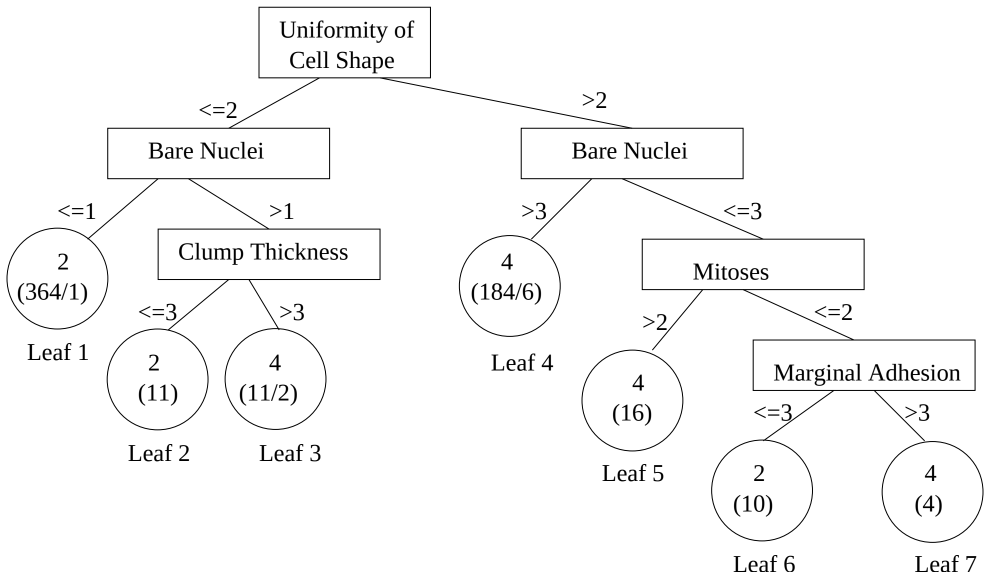

Figure 7.15: The decision tree obtained from a data set perturbed by the *Extended Framework*.

$DT_{ExtndFrm}1$ and $DT_{ExtndFrm}2$ are presented in Figure 7.15 and Figure 7.16, respectively. We study all five trees obtained from the data sets perturbed by *Extended Framework*. None of them has a Type A and a Type D rule. One of them has a Type B rule that applies on 26% records. Most of the trees have Type C rules that apply on 100% records. As a result of the above study we do not consider the trees similar to $DT_{training}$. Therefore, patterns are better preserved in the data sets perturbed by the *Framework* than in the data sets perturbed by the *Extended Framework*. However, the prediction accuracy of the trees perturbed by *Extended Framework* are high. See Table 7.2 for a comparison of the accuracy of these trees to the accuracy of others. All of these perturbed trees also have high accuracy on their underlying data sets. Therefore, these trees represent the underlying data sets well.

The *Extended Framework* is tailored for preserving patterns and correlations among numerical attributes. We compare the correlation matrices of data sets perturbed by the

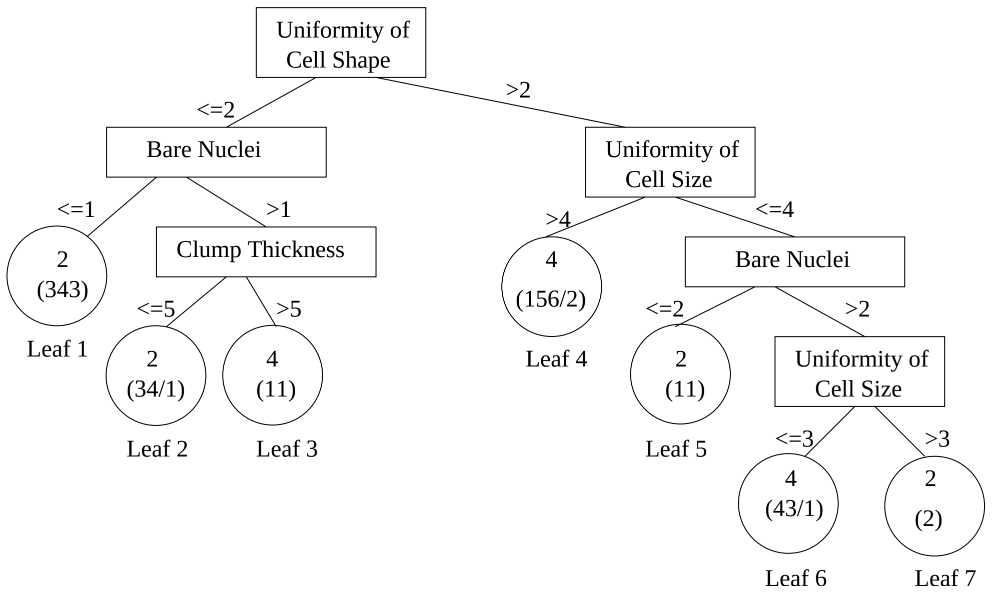

Figure 7.16: The decision tree obtained from a data set perturbed by the *Extended Framework*.

Extended Framework to the correlation matrix obtained from $DT_{training}$. In general, the correlations are better preserved in these data sets than in the data sets perturbed by the *Framework*. The possible reasons include small number of records in a leaf and non multivariate normal distribution of the data set. Correlation matrices of a couple of perturbed data sets are presented as follows.

1) For $DS_{ExtndFrm}1$**:**

Mean vector: [2.94, 2.92, 2.98, 2.85, 2.77, 3.21, 3.13, 2.53, 1.53]

Correlation matrix:

$$\begin{bmatrix}
1.0 & .70 & .71 & .62 & .58 & .70 & .62 & .64 & .40 \\
.70 & 1.0 & .91 & .85 & .76 & .72 & .80 & .74 & .54 \\
.71 & .91 & 1.00 & .81 & .72 & .78 & .80 & .69 & .47 \\
.62 & .85 & .81 & 1.00 & .69 & .73 & .77 & .68 & .45 \\
.58 & .76 & .72 & .69 & 1.00 & .67 & .60 & .75 & .41 \\
.70 & .72 & .78 & .73 & .67 & 1.00 & .74 & .69 & .29 \\
.62 & .80 & .80 & .77 & .60 & .74 & 1.00 & .62 & .38 \\
.64 & .74 & .69 & .68 & .75 & .69 & .62 & 1.00 & .53 \\
.40 & .54 & .47 & .45 & .41 & .29 & .38 & .53 & 1.00
\end{bmatrix}$$

2) For $DS_{ExtndFrm}2$:
Mean vector: [4.81, 2.98, 3.09, 2.52, 2.82, 3.17, 3.04, 2.56, 1.62]
Correlation matrix:

$$\begin{bmatrix}
1.0 & .64 & .66 & .49 & .56 & .55 & .59 & .55 & .42 \\
.64 & 1.0 & .91 & .66 & .82 & .61 & .76 & .70 & .62 \\
.66 & .91 & 1.00 & .65 & .77 & .64 & .75 & .69 & .54 \\
.49 & .66 & .65 & 1.00 & .61 & .67 & .65 & .61 & .42 \\
.56 & .82 & .77 & .61 & 1.00 & .57 & .70 & .67 & .64 \\
.55 & .61 & .64 & .67 & .57 & 1.00 & .63 & .58 & .44 \\
.59 & .76 & .75 & .65 & .70 & .63 & 1.00 & .75 & .47 \\
.55 & .70 & .69 & .61 & .67 & .58 & .75 & 1.00 & .58 \\
.42 & .62 & .54 & .42 & .64 & .44 & .47 & .58 & 1.00
\end{bmatrix}$$

Experiments on the Random Extended-Framework

We experiment on *Random Extended Framework* that applies *GADP* on all records at a time - rather than on the records belonging to a leaf separately. *Random Extended Framework* does not preserve the range of an attribute for the records belonging to a leaf. *Random Extended Framework* also incorporates *ALPT* technique to perturb the class attribute.

We apply *Random Extended Framework* on $DS_{training}$ and produce a perturbed data set. We run this experiment 5 times and produce 5 perturbed data sets $DS_{RandomExt}i$, $1 \leq i \leq 5$. We build the decision tree $DT_{RandomExt}i$ from a perturbed data set $DS_{RandomExt}i$. The sizes (number of leaves) of the perturbed trees are 31, 26, 10, 34 and 39. None of the perturbed *WBC* trees presented so far has such a big size when the size of $DT_{training}$ is 14. Three of the five trees have Type D rules that apply on more than 10% of the records. None of the perturbed trees have a Type A or Type B rule. For the above reasons these trees are considered significantly different from $DT_{training}$ suggesting a noteworthy quality drop in the perturbed data sets.

The prediction accuracy of the perturbed trees are also worse than the accuracy of $DT_{training}$ and the trees obtained from the data sets perturbed by the *Extended Framework*. See Table 7.2 for further details. However, the correlations among the numerical attributes are preserved reasonably well. The mean vector for the numerical attributes of $DS_{RandomExt}1$ is [4.71, 3.77, 3.7, 3.31, 3.25, 4.23, 3.54, 3.41, 1.87] and the correlation matrix is as follows.

$$\begin{bmatrix}
1.0 & .68 & .71 & .56 & .58 & .66 & .61 & .58 & .39 \\
.68 & 1.0 & .88 & .69 & .73 & .65 & .76 & .74 & .44 \\
.71 & .88 & 1.00 & .69 & .73 & .69 & .73 & .74 & .43 \\
.56 & .69 & .69 & 1.00 & .62 & .67 & .71 & .65 & .40 \\
.58 & .73 & .73 & .62 & 1.00 & .58 & .63 & .68 & .49 \\
.66 & .65 & .69 & .67 & .58 & 1.00 & .68 & .62 & .33 \\
.61 & .76 & .73 & .71 & .63 & .68 & 1.00 & .69 & .38 \\
.58 & .74 & .74 & .65 & .68 & .62 & .69 & 1.00 & .44 \\
.39 & .44 & .43 & .40 & .49 & .33 & .38 & .44 & 1.00
\end{bmatrix}$$

Experiments on GADP without ALPT

Since the Random Extended-Framework results in trees that are very dissimilar to $DT_{training}$ we run another set of experiments where we apply *GADP* without *ALPT*. We want to make sure that the quality drop in the perturbed data sets are not only due to *ALPT* but also *GADP*. We perturb $DS_{training}$ by *GADP* on the whole data set at a time instead of on the records belonging to each leaf separately. However, we do not perturb the class attribute. We run this experiment 5 times and thus produce 5 perturbed data sets $DS_{GADP}i$, $i \leq i \leq 5$. We produce 5 decision trees $DT_{GADP}i$ from $DS_{GADP}i$, $1 \leq i \leq 5$.

Similar to the results of our experiments on Random Extended Framework, these decision trees are also very different from $DT_{training}$. All perturbed trees except one have exceptionally big numbers of leaves. The sizes (number of leaves) of the perturbed trees are 31, 26, 10, 34 and 37, respectively. Four out of the five trees have Type D rules that apply on at least 10% of the records. None of the trees has a Type A or Type B rule. The prediction accuracy of $DT_{GADP}1$ and $DT_{GADP}2$ on $DS_{training}$ and $DS_{testing}$ is also lower than the prediction accuracy of $DT_{ExtndFrm}1$ and $DT_{ExtndFrm}2$. The prediction accuracy of these trees is presented in Table 7.3. For the above reasons we consider that the perturbed trees are significantly different from $DT_{training}$ suggesting again a noteworthy quality drop in the perturbed data sets. This result suggests an advantage of our *Extended Framework* where we apply *GADP* on the records belonging to each leaf separately and maintain the ranges of attribute values within each leaf.

Concluding Comments on Experimental Results

Our experimental results indicate that the *Framework* is effective in preserving original patterns and prediction accuracy in perturbed data sets. It does not preserve the original correlations in perturbed data sets. However, the *Extended Framework* preserves original correlations better than the *Framework*. However, it does not preserve original patterns and prediction accuracies as much as the *Framework* does. Both *Framework* and *Extended Framework* preserves data quality better than two other techniques called *Random Framework* and *Random Extended Framework*. Application of *GADP* on the records belonging to each leaf separately according to our *Extended Framework* preserves original patterns better than the use of *GADP* on the whole data set at a time.

7.4 Conclusion

In this chapter we have introduced a *Framework*, that incorporates several techniques to perturb all attributes of a data set. Results of our experiments on *Framework* are very encouraging. They 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. Moreover, the prediction accuracy of the classifier obtained from a data set perturbed by the *Framework* is essentially as high as the accuracy of the original classifier. These experimental results suggest that the *Framework* preserves a high data quality in a perturbed data set, thus allowing quality data mining.

In order to also preserve the original correlations in a perturbed data set we presented an *Extended Framework* that incorporates either *GADP* [73], *C-GADP* [92] or *EGADP* [74]. We have presented results on our experiments with the *Extended Framework*, where we have used *GADP*. Our experimental results suggest that the *Extended Framework* has preserved original correlations well in a perturbed data set. *Extended Framework* has preserved the correlations significantly better than the correlations preserved by other techniques. However, we believe that the *Extended Framework* can preserve original correlations even better if *C-GADP* or *EGADP* is used instead of *GADP* for the following reasons. *GADP* preserves the correlations the best when it is applied on a sufficiently large data set having multivariate normal distribution. The data sets we have used do not have multivariate normal distribution. Moreover, in our *Extended Framework* *GADP* has been applied on small sets of records belonging to the leaves separately. Our future research plans include experiments on the *Extended Framework* using *C-GADP* and *EGADP* instead of *GADP*.

The *Extended Framework* has also preserved a high data quality in terms of prediction accuracy of the classifiers obtained from perturbed data sets. Additionally, it preserves the original patterns significantly better than the patterns preserved by *Random Framework* and *Random Extended Framework*. However, our *Framework* preserves original patterns the better than the patterns preserved by *Extended Framework*. *Extended Framework* has preserved the original patterns reasonably well in the experiments on *Adult* data set. In the experiments on *WBC* data set the decision trees obtained from an original and a perturbed data set are less similar.

GADP regenerates the data from an original correlation matrix. It appears that *GADP* preserves the original correlations but adds a huge amount of noise since it regenerates the data set instead of adding noise to the original data set. Amount of noise addition appears to be high during the regeneration of data. Therefore, *Extended Framework* adds more noise than *Framework*.

However, both *Framework* and *Extended Framework* preserves the domain (defined by the conditional values of the *LINFA*) of each *LINFA* within a horizontal segment resulting in a protection of all original logic rules in a perturbed data set. We confirm the protection of original logic rules by applying the classifier (obtained from an original data set) on a perturbed data set and on the original data set, and getting exactly the same prediction accuracy in both cases. Hence, an obvious question is how a decision tree obtained from a data set perturbed by *Framework* and *Extended Framework* can be different to the original tree. We offer an explanation to this question as follows.

An information gain based decision tree builder algorithm chooses the attribute, for a node of the tree, having the maximum information gain [85]. Since the domain of a *LINFA* is preserved by the *Extended Framework* the information gain of the *LINFA* remains the same in the perturbed data set. However, due to the noise addition the information gain of another attribute can increase and thereby can be greater than the information gain of the original *LINFA*. If this happens then the second attribute would be chosen as the *LINFA* of the tree obtained from the perturbed data set. As a result we will have a perturbed decision tree which will be different from the original tree.

We now explain how the information gain of an attribute can increase due to a noise addition. Suppose a numerical attribute A has the domain [1,10] and the class attribute C has the domain {yes, no} (see Figure 7.17). Let us consider a data set having 20 records, out of which the first group of 10 records (record 1 through to record 10) have $A < 5$ and the second group of 10 records have $A \geq 5$. Also consider that out of the first 10 records 6 records (2, 3, 4, 6, 7 and 9) have $C = yes$ and the remaining 4 records (1, 5, 8 and 10) have $C = no$. Out of the second 10 records 4 records (13, 17, 19 and 20) have $C = yes$ and the remaining 6 records (11, 12, 14, 15, 16 and 18) have $C = no$. Assume that due to the noise addition the value of A for 3 records (5, 8 and 9), originally having $A < 5$ and $C = no$, increases to ≥ 5 . Also consider that the value of A for 3 other records (17, 19 and 20), originally having $A \geq 5$ and $C = yes$, decreases to < 5 (See Figure 7.17). Therefore, due to the noise addition distributions of class values for records having $A < 5$ will change. Similarly, the distribution of class values, for records having $A \geq 5$, will also change. The changes of the distributions will result in an decrease of the entropy for attribute A meaning an increase of the information gain for the attribute.

Both *Framework* and Extended *Framework* preserves a data quality better than two other techniques called Random Framework and Random Extended Framework. Moreover, the application of *GADP* on the records belonging to each leaf separately according to

A	Records	C
<5	2, 3, 4, 6, 7, 9	yes;
	1, 5, 8, 10	no;
>=5	13, 17, 19, 20	yes;
	11, 12, 14, 15, 16, 18	no;

BEFORE PERTURBATION

A	Records	C
<5	2, 3, 4, 6, 7, 9, 17, 19, 20	yes;
	1	no;
>=5	13	yes;
	5, 8, 10, 11, 12, 14, 15, 16, 18	no;

AFTER PERTURBATION

Figure 7.17: An example of how the information gain of an attribute can increase due to a noise addition.

our *Extended Framework* preserves original patterns 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.

Crucial properties of a noise addition technique are the ability to maintain good data quality and to ensure individual privacy. In order to evaluate a perturbation technique we need to measure its ability to preserve high data quality and to provide enough security/protection against privacy breach. In this chapter we have also used our techniques for measuring data quality of a perturbed data set. In the next chapter we present a technique for measuring security level in a perturbed data set.

Classifier Name	Applied On		
	A 25,600 records	B 25,600 records	C 4,562 records
$DT_{training}$ (shown in Figure 7.1)	Corr.: 21,763 Incorr.: 3,837 — 85%	Corr. : 21,763 Incorr.: 3,837 — 85%	Corr. : 3822 Incorr.: 740 — 83.8%
$DT_{Frmwrk}1$ (shown in Figure 7.2)	Corr.: 21,716 Incorr.: 3,884 — 84.8%	Corr. : 21,721 Incorr.: 3,879 — 84.8%	Corr. : 3811 Incorr.: 751 — 83.5%
$DT_{Frmwrk}2$ (shown in Figure 7.3)	Corr.: 21,716 Incorr.: 3,884 — 84.8%	Corr. : 21,703 Incorr.: 3,897 — 84.8%	Corr. : 3806 Incorr.: 756 — 83.4%
$DT_{Random}1$ (shown in Figure 7.4)	Corr.: 15,895 Incorr.: 9,705 — 62.1%	Corr. : 20,739 Incorr.: 4,861 — 81.0%	Corr. : 3,661 Incorr.: 901 — 80.2%
$DT_{Random}2$ (shown in Figure 7.5)	Corr.: 15,968 Incorr.: 9,632 — 62.4%	Corr.: 20,894 Incorr.: 4,706 — 81.6%	Corr. : 3,687 Incorr.: 875 — 80.8%
$DT_{ExtndFrm}1$ (shown in Figure 7.6)	Corr.: 21,677 Incorr.: 3,923 — 84.7%	Corr. : 20,998 Incorr.: 4,602 — 82%	Corr. : 3,717 Incorr.: 845 — 81.5%
$DT_{ExtndFrm}2$ (shown in Figure 7.7)	Corr.: 21,593 Incorr.: 4,007 — 84.3%	Corr. : 21,201 Incorr.: 4,399 — 82.8%	Corr. : 3,728 Incorr.: 834 — 81.7%
$DT_{RandomExt}1$ (shown in Figure 7.8)	Corr.: 16,021 Incorr.: 9,579 — 62.6%	Corr. : 20,396 Incorr.: 5,204 — 79.7%	Corr. : 3,586 Incorr.: 976 — 78.6%
$DT_{RandomExt}2$ (shown in Figure 7.9)	Corr.: 16,041 Incorr.: 9,559 — 62.7%	Corr. : 20,273 Incorr.: 5,327 — 79%	Corr. : 3,592 Incorr.: 970 — 78.7%

Table 7.1: Prediction Accuracy of the Classifiers Obtained from the Unperturbed and Various Perturbed Adult Data Sets.

Classifier Name	Applied On		
	A 600 records	B 600 records	C 83 records
$DT_{training}$ (shown in Figure 7.10)	Corr.: 596 Incorr.: 4 ——— 99.3%	Corr. : 596 Incorr.: 4 ——— 99.3%	Corr. : 74 Incorr.: 9 ——— 89.2%
$DT_{Frmwrk}1$ (shown in Figure 7.11)	Corr.: 596 Incorr.: 4 ——— 99.3%	Corr. : 592 Incorr.: 8 ——— 98.7%	Corr. : 72 Incorr.: 11 ——— 86.8%
$DT_{Frmwrk}2$ (shown in Figure 7.12)	Corr.: 593 Incorr.: 7 ——— 98.8%	Corr. : 590 Incorr.: 10 ——— 98.3%	Corr. : 73 Incorr.: 10 ——— 88%
$DT_{Random}1$ (shown in Figure 7.13)	Corr.: 412 Incorr.: 188 ——— 68.7%	Corr. : 399 Incorr.: 201 ——— 66.5%	Corr. : 55 Incorr.: 28 ——— 66.3%
$DT_{Random}2$ (shown in Figure 7.14)	Corr.: 423 Incorr.: 177 ——— 70.5%	Corr.: 404 Incorr.: 196 ——— 67.3%	Corr. : 53 Incorr.: 30 ——— 63.9%
$DT_{ExtndFrm}1$ (shown in Figure 7.15)	Corr.: 591 Incorr.: 9 ——— 98.5%	Corr. : 576 Incorr.: 24 ——— 96%	Corr. : 77 Incorr.: 6 ——— 92.8%
$DT_{ExtndFrm}2$ (shown in Figure 7.16)	Corr.: 596 Incorr.: 4 ——— 99.3%	Corr.: 568 Incorr.: 32 ——— 94.7%	Corr. : 76 Incorr.: 7 ——— 91.6%
$DT_{RandomExt}1$ (shown in Figure 7.10)	Corr.: 548 Incorr.: 52 ——— 91.3%	Corr. : 565 Incorr.: 35 ——— 94.2%	Corr. : 69 Incorr.: 14 ——— 83.1%
$DT_{RandomExt}2$ (shown in Figure 7.10)	Corr.: 502 Incorr.: 98 ——— 83.7%	Corr.: 532 Incorr.: 68 ——— 88.7%	Corr. : 68 Incorr.: 15 ——— 81.9%

Table 7.2: Prediction Accuracy of the Classifiers Obtained from the Unperturbed and Various Perturbed WBC Data Sets.

Classifier Name	Applied On		
	A 600 records	B 600 records	C 83 records
$DT_{GADP}1$ (without ALPT)	Corr.: 548 Incorr.: 52 ——— 91.3%	Corr.: 565 Incorr.: 35 ——— 94.2%	Corr. : 69 Incorr.: 14 ——— 83%
$DT_{GADP}2$ (without ALPT)	Corr.: 502 Incorr.: 98 ——— 83.7%	Corr.: 532 Incorr.: 68 ——— 88.7%	Corr. : 68 Incorr.: 15 ——— 82%

Table 7.3: Prediction Accuracy of the Classifiers Obtained from WBC Data Sets Perturbed by GADP technique only.

Chapter 8

Measuring of Disclosure Risk

Before we can proceed to security analysis, we need to produce a definition of disclosure. We note that an exposure of any sensitive information can be considered as disclosure. For example, sometimes a set of rules obtained from a data set is considered sensitive and therefore the exposure of the rules is regarded as disclosure [37, 90]. A system proposed in [37, 90] alerts a data miner to sensitive rules, since the miner may not be aware of the sensitivity level of a rule. In another scenario an exposure of any single value is considered as disclosure [3]. Generally, revealing a sensitive attribute value belonging to an individual is considered as disclosure [73]. Such disclosure usually occurs through a re-identification of the record. However, even a re-identification that does not cause an exposure of a sensitive attribute value can still be considered as disclosure [62]. Similarly, an exposure of a sensitive attribute value, without a re-identification, is also regarded as disclosure known as *attribute disclosure* [62]. If an intruder can narrow down the list of all possible records that could have originated from the "target record" to a set of records having the same sensitive attribute value, then the attribute value is considered to be disclosed, even without any exact record re-identification. By "target record" we mean an original record, in which the intruder is interested.

Due to the varying definitions of disclosure it is not trivial to measure disclosure risk. Moreover, 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.

In order to measure disclosure risk Lambert considered the re-identification of a record as disclosure and the resultant exposure of a sensitive attribute value as the harm of the disclosure [62]. Lambert assumes that for every perturbed record an intruder estimates the probability of the record coming from a target record. The intruder then obtains a sensitive attribute value from the perturbed record having the maximum probability. When an intruder believes that he/she has worked out a sensitive information then, regardless of whether or not the intruder has obtained a correct information, Lambert considers this as "perceived disclosure". However, when an intruder obtains a true information then this is considered as "true disclosure".

In what follows we consider both re-identification of the target record and disclosing a confidential class as disclosure and we evaluate each one of them separately.

8.1 Measuring Disclosure Risk

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 disclosure of confidential individual class values contained in the data set; we achieve this not only by perturbing the values of the class attribute itself but also by introducing perturbation to other (non-confidential) attributes, in order to make re-identification of the records difficult and in some instances even impossible;
2. to preserve not only statistical parameters of the data set (means, variances, etc), but also the patterns discovered by the decision tree builder prior to perturbing the data set. We note that there has been a research effort published in the literature where the goal has been to hide confidential patterns [37, 90], but that is beyond the scope of our study.

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 intend to disclose. That might be all of the non-class attributes, some of the attributes or perhaps just one or two of them. 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 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 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 was 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 option 2, which is based on re-identification of the records. We need to show that the intruder is not likely to learn the confidential class through re-identification of the record with more certainty then he/she can learn by running the record through the decision tree. Indeed, we argue that the predictive accuracy of the decision tree built on the data set perturbed by our method is very high as illustrated by our experiments. We shall next discuss the accuracy of predicting the class through 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.

We assume that an original data set having n records and m attributes is perturbed by a technique such as the *Framework*, 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-identfication, 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.

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

where

P_{xi} is the probability of 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^j_{xi}. This probability can be calculated 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 contain the target record x. To illustrate this point, we consider the decision tree in Figure 7.10 obtained from the original *WBC* database. Each attribute has the same domain [1,10]. We select the record with ID=1321264 belonging to the Leaf 1. The logic rule corresponding to this leaf is "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 remain in the range [1,3].

We assume 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^j_{xi} is to run the target record through the decision tree built on the perturbed data set and adopt the attribute domains of the leaf in which the target record arrives. Using this strategy the intruder can calculate the probability P_{xi} for each and every record of the perturbed data set. We use these probabilities to calculate the entropy of the whole perturbed data set with respect to re-identification of a target record x as follows:

$$H^{RI}_x = -\sum_{i=1}^{n} P_{xi} log_2 P_{xi} = -\sum_{i=1}^{n} \left(\frac{\prod_{j=1}^{m} P^j_{xi}}{\sum_{i=1}^{n}(\prod_{j=1}^{m} P^j_{xi})}\right) log_2 \left(\frac{\prod_{j=1}^{m} P^j_{xi}}{\sum_{i=1}^{n}(\prod_{j=1}^{m} P^j_{xi})}\right).$$

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^{RI}_x, $x = 1, 2, \ldots$ ... 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, $\mu = \frac{\sum_{x=1}^{n} H^{RI}_x}{n}$ and the standard deviation, $\sigma = \sqrt{\frac{\sum_{x=1}^{n}(H^{RI}_x - \mu)^2}{n}}$.

The higher entropy H^{RI}_x 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 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.

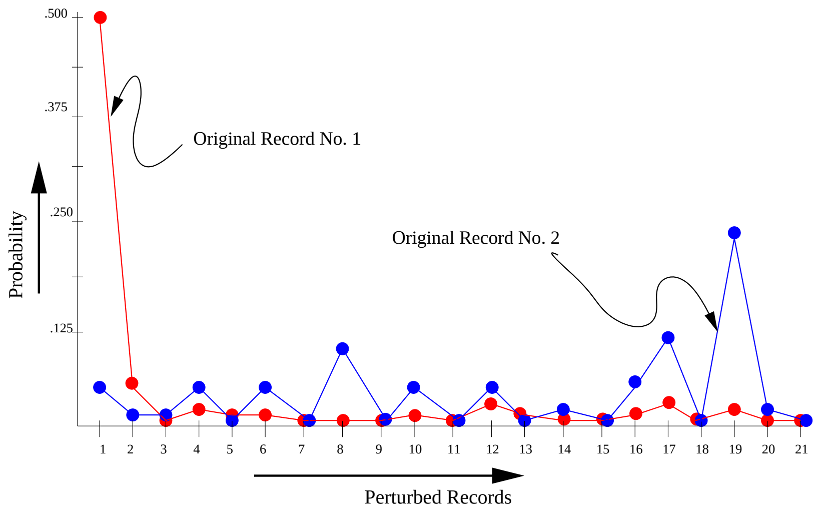

Figure 8.1: The probability distribution of a perturbed record originating from the target record x.

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 8.1 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 8.2). 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}. Therefore, the perturbed data set would not be considered secure for $H_t^{RI} = 2.45$ and $v = 0.05$.

We next explore the risk of an intruder learning the class of the target record. We consider a generalised scenario where an intruder is interested to check if an individual has

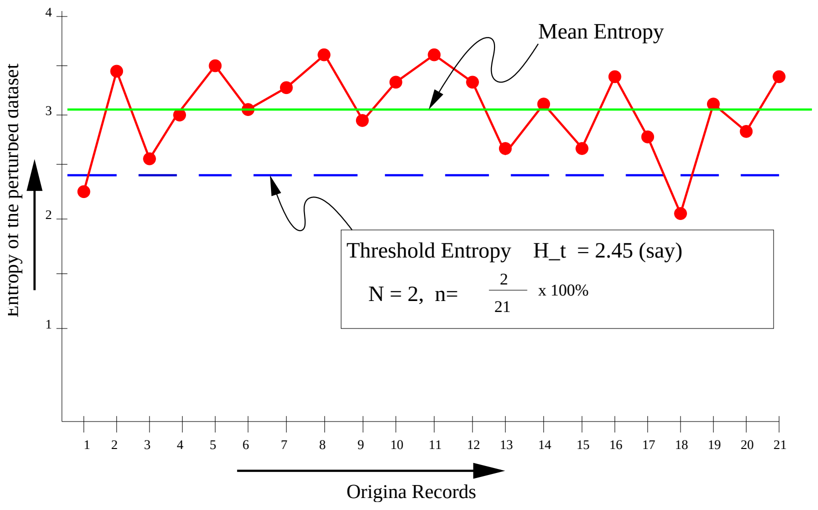

Figure 8.2: Entropies of a perturbed data set calculated for each original record.

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 a 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^c_{xk} 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.

$$P^c_x(L) = \sum_{i=1}^{n}\left(\frac{\prod_{j=1}^{m} P^j_{xi}}{\sum_{i=1}^{n}(\prod_{j=1}^{m} P^j_{xi})} \times \sum_{k=l_1}^{l_z} P^c_i(k)\right)$$

where $P^c_i(k)$ 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 $P^c_i(k)$ 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 this is just the best estimate the intruder can make, as the original and perturbed decision tree may differ to some extent.

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

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

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.

We next turn our attention to the supplementary knowledge an intruder has about the target record x and how important this knowledge is for the successful re-identification of the target record and/or learning its class. As we pointed out earlier the intruders supplementary knowledge can range from knowing all the attributes except for the confidential class attribute, through knowing several attributes including influential ones, to knowing only a few innocent attributes. It is easy to see that the more knowledge the intruder has the more successful he/she will be in re-identifying the record and/or learning the class.

We illustrate this on one of our perturbed *WBC* data set $DS_{Frmwrk}1$. The decision tree built from this data set (shown in Figure 7.11) is very similar to the original decision tree (95.5 % records having Type A rules). This high level of similarity is very typical for the *WBC* data set that is perturbed by our Framework and it will be of assistance to the intruder, thus our estimate will be conservative. We randomly selected a record with ID 1321264 and we calculated both $H_{1321264}^{RI}$ and $H_{1321264}^{c}$. In order to evaluate the results we compare them with the corresponding entropies for the case where the intruder has access to the unperturbed data set. In Table 8.1 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.

The following data is relevant for the purpose of understanding the results in Table 8.1: the total number of records is 600, and for the case where the intruder has no knowledge

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 8.1: A Compare of Entropies for the Cases Where the Intruder Has Access to the Original and the Perturbed Data Set.

about the record except that it exists in the perturbed data set the class entropy $H^c_{1321264} = 0.931$ and the record re-identification entropy $H^{RI}_{1321264} = 9.229$, as shown in the bottom row of the table. The noise added to the attribute values roughly follows Gaussian distribution with mean value 0 and standard deviation 33.33 % of the attribute domain which is [1,10] for leaf innocent attributes and determined by the leaf for Leaf Influential Attributes. Record with ID 1321264 belongs to leaf 1 which has domain $D_2 = [1, 2]$, $D_2 = [1, 3]$ and all the other domains are [1,10].

From Table 8.1 we see that for unperturbed data set the intruder can precisely learn the class whenever he/she knows at least two attribute values for the target record, and can 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 how many attribute values the intruder knows. For example, even if 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. On the other hand, in this example the class entropy remains relatively low whenever intruder knows 3 or more attribute values of the target record.

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. As an extreme case, we can add noise uniformly distributed over whole domain to all innocent attributes and still preserve the patterns. Then from the point of view of an intruder who knows only the values of innocent attributes, all perturbed records will be equally likely and the entropy will be maximum. 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.

In order for perturbed data set to be useful beyond classification it is desirable to keep the noise level low. In that case our method works better for dense data sets such as *WBC*, than for very sparse data sets where preventing re-identification would require higher level of noise simply because the records are very diverse.

8.2 Conclusion

In this chapter we have presented a novel techniques for measuring a disclosure risk of a perturbed data set. Using the technique we can verify a perturbed data set whether or not it is secure. In the next chapter we present an observation on prediction accuracy of a perturbed data set as an indicator of data quality.

Chapter 9

Data Quality

We discussed a number of existing data perturbation techniques in Chapter 3. We also presented novel data perturbation techniques in Chapter 4, Chapter 5, Chapter 6 and Chapter 7. Effectiveness of such techniques is typically evaluated by measuring the disclosure risk and data quality of a perturbed data set. Therefore, we presented a couple of techniques for measuring disclosure risk in Chapter 8. The data quality of a perturbed data set is measured, in Chapter 7, through a few quality indicators. The indicators are the similarity of decision trees built from an original and the perturbed data set, prediction accuracies on underlying data sets and on a testing data set, and the correlation matrices of the original and the perturbed data sets. However, the data quality of a perturbed data set is typically evaluated by just the prediction accuracy of the classifier obtained from the data set [60, 69, 116]. Hence, in this chapter we explore the suitability of prediction accuracy as a sole indicator of data quality. In the experiments we evaluate a data quality by comparing the prediction accuracies of decision trees and neural networks, built from original and perturbed data sets. We then compare this evaluation technique to the one that uses logic rules associated with the decision tree classifiers.

9.1 Motivation

Many studies pay substantial attention to predictive accuracy of a classifier. This motivates us to explore the relationship between the predictive accuracy of a classifier and the data quality of the underlying data set on which the classifier was built.

Lim, Loh and Shih [69] used predictive accuracy of classifiers in order to compare them.

They used twenty-two decision trees, nine statistical, and two neural network classifiers in their experiments. They worked with sixteen data sets and added independent noise to each of them and thus they got sixteen perturbed data sets. They compared all classifiers on these 16 original and 16 perturbed data sets in terms of classification error rate and computational time. Additionally, for decision tree classifiers they used the number of nodes as a base for comparison. They found that the average error rates for the majority of classifiers are not statistically significant. All but three classifiers adjusted to noise well. The performances of the classifiers were rated good, average or bad. None of the decision trees or statistical classifiers had a good performance on the unperturbed data set when it had a bad performance on the corresponding perturbed data set. This was the case for all 16 data sets. One of the two neural network classifiers was rated “good” for an original data set and “bad” for the corresponding perturbed data set. However, this was the case for only one out of all 16 data sets.

Lim, Loh and Shih [69] found that despite of independent noise addition the prediction accuracy of a decision tree built on a perturbed data set always remains good when the accuracy of the tree built on the original data set is good. This is understandable as independent noise will not significantly affect homogeneous leaves. However, it is also well known that the structure of decision trees is indeed very sensitive to noise [64] and some patterns especially less prominent ones will be destroyed by noise. Keeping in mind that one of the main purposes of data mining is exploring new patterns, we argue that the finding of Lim, Loh and Shih exposes the fact that a good prediction accuracy may not be always a good measure of the underlying data quality.

Wilson and Rosen [116] compared several perturbation techniques, namely Simple Additive Data Perturbation (SADP), General Additive Data Perturbation (GADP) excluding the categorical dependant variable, and General Additive Data Perturbation (GADP-W) including the categorical dependant variable, in terms of predictive accuracies of classifiers built on the perturbed data sets. General Additive Data Perturbation technique was originally proposed by Muralidhar, Parsa and Sarathy [73]. They also introduced four types of bias and showed that previously proposed perturbation techniques, including SADP, experienced at least one of these biases. They then showed that General Additive Data Perturbation technique was free of these four biases. Wilson and Rosen [116] commented that biases discussed by Muralidhar et al. [73] did not include a possible bias that could be generated from changes of knowledge-based relationship. They called such bias “type

DM" bias. They used 2 different data sets, IRIS Plant Database and BUPA Liver Disorders Database. SPSS's Answer Tree software, using QUEST method, was used as the classifier. For the Liver database, there was no significant difference between the predictive accuracy of the data sets perturbed by different perturbation techniques and the original data set. However, for the IRIS database the performance of GADP was significantly worse than even that of SADP. Wilson and Rosen suggested that this was an evidence of existence of Type DM bias. Wilson and Rosen used the prediction accuracy as a measure of data quality, in their experiments to compare the performances of different perturbation techniques.

Kohavi [60] found that the predictive accuracy of a Naive-Bayes classifier is better for small data sets and the predictive accuracy of Decision Tree is better for large data sets. He then proposed a classifier that was a hybrid of Naive-Bayes and Decision Tree classifiers.

Agrawal and Srikant [3] argued that the extraction of aggregate data was the main task of data mining. They attempted to develop accurate models without knowing individual records. They trained the decision tree classifier on the dataset that had been perturbed. They used the perturbed distribution to reconstruct the distribution of original dataset using a novel reconstruction technique. Then they built decision tree classifiers from these reconstructed distributions and compared their predictive accuracies with the predictive accuracy of the decision tree built from the original data.

9.2 Our Work

In this chapter we concentrate on the predictive accuracy of classifiers obtained from original and several carefully perturbed data sets. We use two different data sets, namely *Boston Housing Price (BHP)* data set having 350 records, and *Wisconsin Breast Cancer (WBC)* data set having 699 records. The *WBC* data set has been introduced in Chapter 4. Both of these data sets are available from the UCI Machine Learning Repository [77]. They are very often used in the data mining community for various research purposes. We divide each of these data sets into two parts, the training set and the testing set. The *BHP* data set is thus divided into 300 records of training set and 50 records of testing set. Similarly the *WBC* data set is divided into 350 records of a training set and 350 records of a testing set with one record overlapping.

We first build a classifier, using well-known Quinlan's decision tree builder See5, from the training set. A decision tree obtained from the training *BHP* data set is shown in

Figure 9.1). We then perturb the training data set using three class attribute perturbation techniques, namely *Random Perturbation Technique (RPT)*, *Probabilistic Perturbation Technique (PPT)* and *All Leaves Probabilistic Perturbation Technique (ALPT)* introduced in Chapter 4. The expected number of changed records is equal in all of the three perturbation techniques.

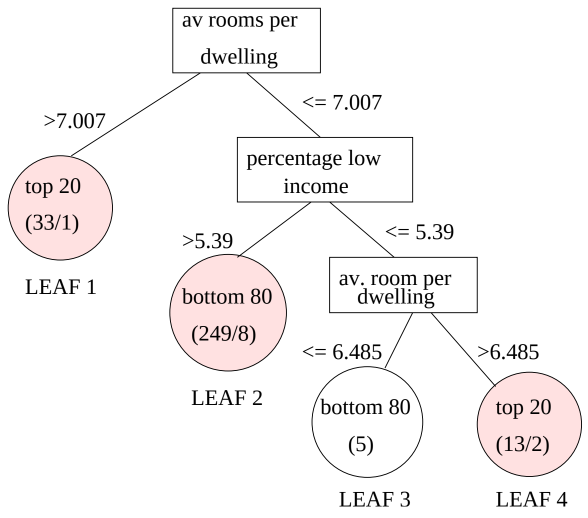

Figure 9.1: A decision tree obtained from the training *BHP* data set having 300 records. Squares represent internal nodes, unshaded circle represents homogeneous leaf and shaded circles represent heterogeneous leaves.

We then use See5 Decision Tree builder (produced by RuleQuest Research) for building decision trees from perturbed data sets. We first perturb an original data set applying RPT and build a decision tree classifier from the perturbed data set. We then apply the classifier on the testing data set, the original data set and the perturbed data set and record the prediction accuracy of the classifier on each of the three data sets. We repeat the test a few times. We then carry out similar tests for data sets perturbed by PPT and $ALPT$. We use both BHP and WBC data set for all experiments.

We then build a Neural Network classifier from an original data set. We apply the classifier on the testing data set and the original data set and record the prediction accuracies. We then perturb the original data set by RPT and build a Neural Network classifier from the perturbed data set. We apply the classifier to the testing data set, the original data set and the perturbed data set from where the classifier is built. We perturb the original data set by RPT four more times and repeat the same experiment each time. We carry out similar experiments on data sets perturbed by *PPT* and *ALPT.* We carry out 5 tests on each of the techniques. In the experiments with Neural Network classifier we use both *BHP* and *WBC* data set for all tests. We present our experimental result in the next section.

For Neural Network classifier we use STATISTICA Neural Network software of StatSoft Inc. and three layer network architecture, namely the input layer, the hidden layer/s and the output layer, using back propagation training algorithm for all experiments carried out. We now describe the Neural Network used in experiments on a WBC data set perturbed by *Probabilistic Perturbation Technique (PPT).* The input layer has eight input variables, each of the two hidden layers has four neurons and the output layer has one output variable. Weights and threshold values are shown below, where H1.2 means neuron number 2 of hidden layer number 1.

```
          H1.1    H1.2    H1.3    H1.4     H2.1   H2.2     H2.3     H2.4
Thresh. -0.8735 -1.5528 -1.60359 0.614384 2.9101 1.297828 2.227079 -0.4534
VAR2    -0.9523  0.2020 -1.0496  0.654275
VAR3    -0.1831 -0.9349 -1.46283 1.624865
VAR4    -0.8872 -1.5113 -1.47357 0.3525004
VAR5    -1.1136 -0.8607 -0.01692 1.027643
VAR6    -1.4474 -2.2931 -2.67157 1.620856
VAR7    -0.7612 -1.5256 -1.90809 0.3553013
VAR8     0.6426  0.2769 -0.6655  0.7896174
VAR9    -0.2887 -1.8095 -1.39306 0.8494807
H1.1                                      -0.213   1.56727 -1.14158 -0.6792
H1.2                                      -1.35024 2.62223 -0.5001  -3.1151
H1.3                                      -0.6733  3.37780 -1.72689 -2.9908
H1.4                                      -0.8417 -2.18433 -1.39512  2.9166
```

9.3 Experimental Results

We present our experimental results in four tables. Table 9.1 and Table 9.2 show the experimental results of Decision Tree (DT) classifiers on *BHP* and *WBC* data set respectively. In these tables DT1 and DT8 are obtained from original data sets while DT2, DT3 and DT9 are obtained from data sets perturbed by *Random Perturbation Technique (RPT).* DT4, DT5 and DT10 are obtained from data sets perturbed by *Probabilistic Perturbation Technique (PPT)* while the rest DTs of Table 9.1 and Table 9.1 are obtained from data sets perturbed by *All Leaves Probabilistic Perturbation Technique (ALPT).*

Name of Decision Tree	Number of Attr. in the Tree	Attr. Both in Original and This Tree	Attr. in this but Not in the Original Tree	Attr. in Original Tree but Not in this Tree	Tree Size	Data set	% of Incorrect Prediction
A	B	C	D	E	F	G	H
DT 1 Original	4	-	-	-	4	Original	3
						Testing	12
DT 2 *RPT*	4	3	1	1	6	Original	3
						Testing	10
						Perturbed	3
DT 3 *RPT*	4	3	1	1	6	Original	4.67
						Testing	8
						Perturbed	3.67
DT 4 *PPT*	2	2	0	2	4	Original	4.67
						Testing	10
						Perturbed	4.33
DT 5 *PPT*	4	3	1	1	6	Original	4
						Testing	12
						Perturbed	2
DT 6 *ALPT*	6	3	3	1	11	Original	3.33
						Testing	12
						Perturbed	5.67
DT 7 *ALPT*	2	2	0	2	3	Original	4.67
						Testing	10
						Perturbed	8.33

Table 9.1: Experimental Results of Decision Tree Classifiers on the BHP Data Set.

The neural network classifier built from the original training BHP data set is called BHP-Original in Table 9.3. We apply this classifier on the testing BHP data set and the original BHP data set.

We perturb the original BHP data set 5 times by *RPT* and build neural network classifiers from each of these 5 perturbed data sets. In general we call these classifiers BHP-Random, which is a set of 5 classifiers. We apply each of these 5 classifiers on the BHP testing data set, the BHP original data set and the perturbed data set from where the classifier is built. In column E, F and G of Table 9.3 we show the mean, median and mode of the prediction accuracies of 5 classifiers of BHP-Random classifier set. Similarly, BHP Probabilistic and BHP-All-Leaves are two sets of 5 classifiers obtained from the data sets perturbed by *PPT* and *ALPT* respectively, while BHP-Original is a single classifier.

Name of Decision Tree	Number of Attr. in the Tree	Attr. Both in Original and This Tree	Attr. in this but Not in the Original Tree	Attr. in Original Tree but Not in this Tree	Tree Size	Data set	% of Incorrect Prediction
A	B	C	D	E	F	G	H
DT 8 Original	4	-	-	-	4	Original	1.71
						Testing	1.71
DT 9 *RPT*	4	4	0	0	6	Original	1.71
						Testing	1.71
						Perturbed	1.71
DT 10 *PPT*	5	4	1	0	7	Original	2.86
						Testing	3.14
						Perturbed	2.86
DT 11 *ALPT*	6	3	3	1	11	Original	1.14
						Testing	0.86
						Perturbed	2.86

Table 9.2: Experimental Results of Decision Tree Classifiers on the WBC Data Set.

Table 9.4 is identical to Table 9.3 except that Table 9.4 deals with WBC data set.

We now compare our experimental results to those that we present in Chapter 4, where the data quality was measured by the similarities of the Decision Trees. The similarity of decision trees (DTs) was measured by the similarity of logic rules associated with the DTs. The data quality of a data set perturbed by the *ALPT* was the worst among these three perturbed data sets, and the data quality of a data set perturbed by *PPT* is slightly worse than that of a data set perturbed by the *RPT*. These results were as expected. *ALPT* technique adds noise randomly to all records in a data set. On the other hand, *RPT* and *PPT* only add noise to records in heterogenous leaves, and in such a way that the total numbers of minority and majority records remain unchanged, or very similar with high probability. Thus these techniques completely preserve all the patterns identified by the leaves of an original decision tree, which are arguably the strongest patterns in the data set.

Our experimental results show that the prediction accuracy of the DT classifiers as a measure of data quality is not consistent with comparing the trees based on logic rules. Careful observation of Table 9.1 and Table 9.2 reveals that the prediction accuracies of classifiers obtained from data sets, which are perturbed by *ALPT*, are not significantly worse than the prediction accuracies of the classifiers obtained from other data sets. However, the

Classifier Name	Classifier Produced From	Classifier Applied on	Mean of % of Error in Prediction	Median of % of Error in Prediction	Mode of % of Error in Prediction
A	B	C	D	E	F
BHP_Original	Original BHP data set	Original	9.42	9.42	9.42
		Testing	8	8	8
BHP_Random	BHP data set, perturbed by *RPT*	Original	11.67	13	Diff
		Testing	14	14	Diff
		Perturbed	12.87	15	Diff
BHP_Probabilistic	BHP data set, perturbed by *PPT*	Original	11.93	12.67	Diff
		Testing	14	16	16
		Perturbed	14.13	15	Diff
BHP_All-Leaves	BHP data set, perturbed by *ALPT*	Original	18.53	19.67	Diff
		Testing	23.2	22	Diff
		Perturbed	20.73	22	Diff

Table 9.3: Experimental Results of Neural Network Classifier on BHP Data Set. “Diff.” in Col. G Means That There are 5 Different Values in All 5 Experiments and Hence There is No Single Mode Value.

Classifier Name	Classifier Produced From	Classifier Applied on	Mean of % of Error in Prediction	Median of % of Error in Prediction	Mode of % of Error in Prediction
A	B	C	D	E	F
WBC_Original	Original WBC data set	Original	2.86	2.86	2.86
		Testing	1.71	1.71	1.71
WBC_Random	WBC data set, perturbed by *RPT*	Original	2.97	2.86	2.29,3.71
		Testing	1.6	1.43	1.43
		Perturbed	3.89	3.43	3.43
WBC_Probabilistic	WBC data set, perturbed by *PPT*	Original	3.03	3.43	3.43
		Testing	2.46	2.29	2
		Perturbed	3.31	3.71	3.71
WBC_All-Leaves	WBC data set, perturbed by *ALPT*	Original	3.49	3.14	3.14
		Testing	2.11	1.71	1.43
		Perturbed	6.57	6.57	7.71

Table 9.4: Experimental Results of Neural Network Classifier on WBC Data Set.

data quality of the data sets, which are perturbed by $ALPT$, appears to be worse than the same of other data sets in the sense of the dissimilarity of the trees (see col. B, C, D, E and F of Table 9.1 and Table 9.2) and also how we perturb the data sets.

Table 9.4 demonstrates that for WBC data set the prediction accuracy of the Neural Network classifier is also not significantly correlated to the data quality of the data set. However, Table 9.3 illustrates that for BHP data set, this is not the case. Here $ALPT$ gives much worse prediction accuracy than the other methods. This is consistent with dissimilarity of the trees as a measure of data quality.

9.4 Conclusion

Our experimental results indicate that measuring data quality by prediction accuracy of the decision trees is inconsistent with measuring data quality by similarity of the decision trees built on original and perturbed data sets. Although, $ALPT$ technique seemed to produce significantly worse results in the sense of decision trees' similarity, it doesn't appear to be worse in the sense of prediction accuracy. The exception to this observation is neural network classifier on BHP data set. It is well known that in terms of prediction accuracy neural networks are more sensitive to noise than decision trees and it appears that techniques such as RPT and PPT are much better suited than $ALPT$ to protect the confidentiality of records in the data sets used for building neural network classifiers. However, these two techniques do not seem to be superior for decision trees, in the terms of prediction accuracy.

We offer a few possible explanations. When we add little amount of noise we drop off the data quality, which is reflected by the dissimilarity of the DTs obtained from the perturbed data sets. However, as added noise is random and does not exhibit any significant pattern it is highly unlikely that a new pattern will be generated in the perturbed data set. It is much more probable that adding the noise will only weaken or strengthen the existing patterns. This is illustrated in Table 9.1 and Table 9.2 where the classifier described in the last row of each table actually performed worse on the perturbed data sets on which they were built than on the original data sets. In other words perturbed data sets did not exhibit as strong patterns as the original data set.

When a decision tree has a good prediction accuracy on the underlying data set (i.e. the data set from which the tree is obtained) then we consider that the tree represents the patterns of the underlying data set very well. Therefore, we suggest that if a decision tree obtained from a perturbed data set is similar to the tree obtained from the original data set and both trees have good accuracies on their underlying data sets then the perturbed data set can be considered as similar to the original data set. Therefore, in such a case the data quality of the perturbed data set can be considered as good.

Chapter 10

Conclusion

We presented a privacy preserving technique that adds noise to each and every attribute, both numerical and categorical, of a data set. We added noise in such a way so that a high data quality is preserved in the perturbed data set. We measured data quality through the following quality indicators: degree of similarity between two decision trees obtained from an original and a perturbed data set, prediction accuracy of the decision trees, and correlation matrices of the original and the perturbed data set. Therefore, the perturbed data set can be used for classification, prediction and correlation analyses. Moreover, since we add a little amount of noise the perturbed data set can also be used for many other data analyses. Since noise is added to all attributes, it makes record re-identification determining the confidential class values difficult.

We presented techniques for adding noise to a sensitive class attribute. We added the same amount of noise in three class attribute perturbation techniques, namely the RPT, PPT and $ALPT$. We compared results of our experiments on all these techniques. Our experimental results suggest that the RPT and PPT preserve the patterns better than the $ALPT$ - although the same amount of noise has been added in the techniques.

Noise addition to a sensitive class attribute reduces the disclosure risk of the confidential class value belonging to an individual. Noise addition to all non-class numerical attributes along with the class attribute further improves the security since an intruder then faces higher level of difficulty in re-identifying a record in the first place. Moreover, some numerical attributes, such as "*salary*" can themselves be sensitive. Therefore, we presented techniques to add noise to all non-class numerical attributes. We divided the non-class numerical attributes into two categories, namely the *Leaf Influential Attribute*

(LINFA) and the *Leaf Innocent Attribute (LINNA)*. We perturbed these attributes by *LINFAPT* and *LINNAPT* respectively. They used noise having normal distribution with mean $\mu = 0$ and variance σ^2. The *LINFAPT* preserves the range defined by the conditional values of a *LINFA*. *LINNAPT* preserves the original domain of a *LINNA*. Therefore, these techniques add noise to all numerical attributes by maintaining all logic rules revealed in an original decision tree. We compared these techniques with a technique called *RNAT* that does not preserve the ranges of *LINFAs* and adds noise having a uniform distribution. Our experimental results indicate that both *LINFAPT* and *LINNAPT* preserve original patterns much better than the *RNAT*.

Many data sets [77] have both numerical and categorical non-class attributes. For the protection of individual privacy in such a data set we perturbed all non-class categorical attributes along with other attributes. However, due to the absence of any natural ordering among categorical values it is not straightforward to add noise to them. Following few existing techniques [40, 11, 61] we first clustered categorical values belonging to an attribute. We used a novel clustering technique called *DETECTIVE* that has a few differences with the existing techniques. Unlike most existing techniques *DETECTIVE* takes numerical attributes as well into account while clustering categorical values. Therefore, *DETECTIVE* is directly applicable to a data set having both numerical and categorical attributes. *DETECTIVE* divides a data set in horizontal segments specific to an attribute for clustering. It discovers the similarity between two values within a horizontal segment as two values may be similar within a horizontal segment while they are not similar in the whole data set. *DETECTIVE* explores similarity among values belonging to an attribute focusing on few other relevant attributes only. We have experimentally compared *DETECTIVE* with an existing technique called *CACTUS* and found *DETECTIVE* suitable for noise addition. We presented *CAPT* a categorical value clustering technique that uses *DETECTIVE.* Our experimental results indicate that *CAPT* preserves the original patterns while adding noise to categorical values.

We presented a *Framework* that incorporates the techniques presented in the thesis to perturb all attributes (both numerical and categorical) of a data set. Our experimental results suggest that a perturbed data set preserves original patterns and prediction accuracies very well. Additionally due to the noise addition to all attributes it protects individual privacy very well. We also presented an *Extended Framework*, which incorporates an existing technique called *GADP* with our categorical attribute and class attribute perturbation techniques. Our experimental results suggest that the data set perturbed by the *Extended Framework* also maintains the original correlations, at the expense of slightly reduced pattern preservation when compared to our original *Framework*. We experimentally compared our *Framework* and *Extended Framework* with two other random perturbation techniques and pointed out the effectiveness of our proposed techniques in preserving good data quality while adding noise. The *Extended Framework* applies *GADP* on horizontal segments of leaves separately and preserves original logic rules in a perturbed data set. Therefore, the *Extended Framework* preserves original patterns better than the *GADP* when it is applied on a whole data set.

In the *Extended Framework* we can use *C-GADP* or *EGADP* instead of *GADP* for possibly maintaining the original correlations better. We also offered an explanation why decision trees obtained from a data set perturbed by the *Extended Framework* are often different from the original decision tree, although all original logic rules are preserved in the perturbed data sets.

We also presented an entropy based technique for measuring the disclosure risk in a perturbed data set.

Typically data quality is evaluated by just the prediction accuracy of the classifier built on a perturbed data set [60, 69, 116]. We assessed this evaluation technique and compared it with the technique/approach that we use for quality evaluation.

We discussed many existing noise addition techniques in Chapter 3. Some of them [73, 92, 74] add noise to numerical attributes only and preserve statistical parameters such as mean and covariance matrix. These techniques do not intend to preserve patterns such as classification rules. Another technique [3] adds random noise to all attributes in such a way that the distribution of data values belonging to an original and a perturbed data set are very different. It is possible to approximately reconstruct original distributions with the help of a proposed technique. However, the original values of individual records can not be precisely estimated from a perturbed data set. Therefore, attribute correlations and patterns of an original data set are neither preserved nor recoverable. The technique proposed by Estivill-Castro and Brankovic [28] preserves patterns, but adds noise only to categorical class attribute. Our technique adds noise to all numerical and categorical attributes and preserves both statistical parameters and data mining patterns such as classification rules. A data set perturbed by the technique maintains high data quality and high level of security.

We believe that future research efforts in this area should focus on developing noise addition techniques applicable to a wider spectrum of data mining tasks, including clustering and mining for association rules. Our future efforts in this direction will start with further refinements of our *Extended Framework*, especially using *CGADP* and *EGADP*, including further experiments.

Another future direction emerging from this thesis is a generalisation of our clustering techniques, that is, *EX-DETECTIVE* in order to make it more flexible regarding the size and compactness of the desired clusters.

We also plan to further improve the data quality measures. For example, it would be good if we can come up with a single quality value based on various quality indicators such as prediction accuracy and similarity. We would also explore if our security measuring technique can be improved.

Bibliography

[1] N. Adam and J. C. Wortmann. Security control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21(4):515–556, 1999.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Santa Barbara, California, USA, May 2001.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[4] S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *Proc. of 21st International Conference on Data Engineering (ICDE 2005), IEEE*, pages 193–204, 2005.

[5] S. Agrawal, V. Krishnan, and J. R. Haritsa. On addressing efficiency concerns in privacy-preserving mining. In *Proc. of 9th International Conference on Database Systems for Advances Applications (DASFAA 2004)*, pages 113–124, Jeju Island, Korea, March 17-19 2004.

[6] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C Sevcik. Clustering categorical data based on information loss minimization. In *Proc. of the 2nd Hellenic Data Management Symposium (HDMS'03)*, Athens, Greece, September 2003.

[7] P. Andritsos, P. Tsaparas, R. J. Miller, and K. C. Sevcik. Limbo: Scalable clustering of categorical data. In *Proc. of the 9th International Conference on Extending Data Base Technology (EDBT)*, Heraklion-Crete, Greece, March 2004.

[8] M. Atallah, E. Bertino, A. Elmagarmid M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proc. of IEEE Knowledge and Data Engineering Workshop*, pages 45–52, Chicago, Illinios, USA, November 1999.

[9] D. Barbara, J. Couto, and Y. Li. Coolcat: An entropy-based algorithm for categorical clustering. In *Proc. of ACM International Conference on Information and Knowledge Management*, 2002.

[10] D. Bentley. Randomized response. available from http://www.dartmouth.edu/ chance/teaching_aids/RResponse/RResponse.html. visited on 12.01.07.

[11] L. Brankovic. *Usability of Secure Statistical Databases.* PhD dissertation, The University of Newcastle, Australia, Newcastle, Australia, 1998.

[12] L. Brankovic and V. Estivill-Castro. Privacy issues in knowledge discovery and data mining. In *Proc. of Australian Institute of Computer Ethics Conference (AICEC99)*, Melbourne, Victoria, Australia, July 1999.

[13] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi. *Discovering Data Mining from Concept to Implementation.* Prentice Hall PTR, New Jersey 07458, USA, 1998.

[14] A. Cavoukian. Data mining: Staking a claim on your privacy, Information and Privacy Commissioner Ontario. *Available from http://www.ipc.on.ca/ docs/datamine.pdf, Accessed on 21 May, 2008*, 1998.

[15] A. Cavoukian. Tag, you're it: Privacy implecations of radio frequency identification (rfid) technology, Information and Privacy Commissioner Ontario. *Available from https://ozone.scholarsportal.info/ bitstream/1873/6228/1/10318697.pdf, Accessed 21 May, 2008*, 2004.

[16] K. Chuang and M. Chen. Clustering categorical data by utilizing the correlated-force ensemble. In *Proc. of the 4th SIAM International Conference on Data Mining (SDM 04)*, April 22-24, 2004.

[17] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.

[18] T. Dalenius and S. P. Reiss. Data-swapping: A technique for disclosure control. *Journal of Statistical Planning and Inference*, 6(1):73–85, 1982.

[19] C. J. Date. *An Introduction to Database Systems.* Addison Wesley, 7th edition, 2000.

[20] S. Datta, H. Kargupta, and K. Sivakumar. Homeland defense, privacy-sensitive data mining, and random value distortion. In *Proc. of the SIAM Workshop on Data Mining for Counter Terrorism and Security (SDM'03)*, May 2003.

[21] P. de Wolf, J. M. Gouweleeuw, P. Kooiman, and L. Willenborg. Reflections on PRAM. In *Conference Programme of Statistical Data Protection '98*, Lisbon, 1998.

[22] R. Delmater and M. Hancock. *Data Mining Explained: A Manager's Guide to Customer-Centric Business Intelligence.* Digital Press, Boston, USA, 2001.

[23] W. Du and M. J. Atallah. Secure multiparty computation problems and their applications: A review and open problems. In *Proc. of New Security Paradigms Workshop*, pages 11–20, Cloudcroft, New Mexico, USA, September 11-13 2001.

[24] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proc. of the Fourth SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, USA, April 22-24 2004.

[25] W. Du and Z. Zhan. Building decision tree classifier on private data. In *Workshop on Privacy, Security, and Data Mining at The 2002 IEEE International Conference on Data Mining (ICDM'02)*, Maebashi City, Japan, December 9 2002.

[26] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 505–510, Washington, DC, USA, August 2003.

[27] M. Dutta, A. K. Mahanta, and A. K. Pujari. Qrock: A quick version of rock algorithm for clustering of categorical data. *Pattern Recognition Letters*, 26(15):2364–2373, 2005.

[28] V. Estivill-Castro and L. Brankovic. Data swapping: Balancing privacy against precision in mining for logic rules. In *Proc. of Data Warehousing and Knowledge Discovery (DaWaK99)*, 1999.

[29] V. Estivill-Castro and I. Lee. Amoeba: Hierarchical clustering based on spatial proximity using delaunaty diagram. In *Proc. of the 9th International Symposium on Spatial Data Handling*, pages 7a.26–7a.41, 2000.

[30] V. Estivill-Castro and I. Lee. Autoclast: Automatic clustering via boundary extraction for mining massive pont-data sets. In *Proc. of the 5th International Conference on Geocomputation*, 2000.

[31] A. V. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 211–222, San Diego, CA, USA, June 2003.

[32] A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, 2002.

[33] C. Farkas and S. Jajodia. The inference problem: A survey. *ACM SIGKDD Explorations Newsletter*, 4(2):6–11, December 2002.

[34] S. E. Fienberg. Privacy and confidentiality in an e-commerce world: Data mining, data warehousing, matching and disclosure limitation. *Statistical Science*, 21:143–154, 2006.

[35] S. E. Fienberg and J. McIntyre. Data swapping: Variations on a theme by Dalenius and Reiss. *Journal of Official Statistics*, 21:309–323, 2005.

[36] American Association for Artificial Intelligence. Fraud detection and prevention. available from http://www.aaai.org/aitopics/html/fraud.html. visited on 07.07.06.

[37] P. Fule and J. F. Roddick. Detecting privacy and ethical sensitivity in data mining results. In Vladimir Estivill-Castro, editor, *Proc. of 27th Australian Computer Science Conference (ACSC2004)*, volume 26 of *Conference in Research and Practice in Information Technology*, pages 163–168, 2004.

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

[39] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. In *Proc. of the 24th VLDB Conference*, New York, USA, 1998.

[40] H. Giggins and L. Brankovic. Protecting privacy in genetic databases. In *Proc. of of the 6th Engineering Mathematics and Applications Conference (EMAC 2003)*, volume 2, pages 73–78, Sydney, Australia, 2003.

[41] B. Gilburd, A. Schuster, and R. Wolff. Privacy-preserving data mining on data grids in the presence of malicious participants. In *Proc. of 13th International Symposium on High-Performance Distributed Computing (HPDC-13 2004)*, pages 225–234, Honolulu, Hawaii, USA, June 2004.

[42] E. Gokcay and J. C. Principe. Information theoretic clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):158–171, February 2002.

[43] J.M. Gouweleeuw, P. Kooiman, L.C.R.J. Willenborg, and P.P. de Wolf. Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14(4):463–478, 1998.

[44] R. Groth. *Data Mining A Hands-On Approach For Business Professionals.* Prentice Hall PTR, New Jersey 07458, USA, 1998.

[45] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proc. of the 15th International Conference on Data Engineering*, Sydney, Australia, March 23-26, 1999.

[46] J. Han. How can data mining help bio-data analysis? In *Proc. of Workshop on Data Mining in Bioinformatics (BIOKDD'02)*, pages 1–4, Edmonton, Canada, 2002.

[47] J. Han and M. Kamber. *Data Mining Concepts and Techniques.* Morgan Kaufmann Publishers, San Diego, CA 92101-4495,USA, 2001.

[48] A. A. Hintoglu and Y. Saygin. Suppressing microdata to prevent probabilistic classification based inference. In *Proc. of Secure Data Management, Second VLDB Workshop, SDM 2005*, pages 155–169, Trondheim, Norway, 2005.

[49] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of SIGKDD'02*, Edmonton, Alberta, Canada, 2002.

[50] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Trans. Knowl. Data Eng.*, 16(9):1026–1037, 2004.

[51] M. Kantarcoglu and J. Vaidya. Privacy preserving naive bayes classifier for horizontally partitioned data. In *Proc. of IEEE ICDM Workshop on Privacy Preserving Data Mining*, pages 3–9, Melbourne, Florida, USA, November 2003.

[52] M. Kantardzic. *Data Mining Concepts, Models, Methods, and Algorithms.* IEEE Press, NJ, USA, 2003.

[53] H. Kargupta, K. Liu S. Datta, J. Ryan, and K. Sivakumar. Link analysis, privacy preservation, and random perturbations. In *Proc. of Workshop on Link Analysis for Detecting Complex Behavior (LinkKDD2003)*, Washington, D.C., USA, 2003.

[54] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 99–106, Melbourne, Florida, USA, December 2003.

[55] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems*, 7:387–414, 2005.

[56] J. J. Kim. A method for limiting disclosure in microdata based on random noise and transformation. In *American Statistical Association, Proc. of the Section on Survey Research Methods*, pages 303–308, 1986.

[57] J. J. Kim. Subpopulation estimation for the masked data. In *American Statistical Association, Proc. of the Section on Survey Research Methods*, pages 456–461, 1990.

[58] J. J. Kim and W. E. Winkler. Masking microdata files. In *American Statistical Association, Proc. of the Section on Survey Research Methods*, pages 114–119, 1995.

[59] J. J. Kim and W. E. Winkler. Multiplicitive noise for masking continuous data. Technical report, Statistical Research Division, U.S. Bureau of the Census, April 17 2003.

[60] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[61] P. Kooiman, L. Willenborg, and J. Gouweleeuw. Pram: A method for disclosure limitation of microdata. Research Paper 9705, Statistics Netharlands, P.O. Box 4000, 2270 JM Voorburg, The Netharlands, June 1997.

[62] D. Lambert. Measures of disclosure risk and harm. *Journal of Official Statistics*, 9:313–331, 1993.

[63] I. Lee and V. Estivill-Castro. Effective and efficient boundary-based clustering for three-dimensional geoinformation studies. In *CODAS*, pages 87–96, 2001.

[64] R. Li. *Instability of Decision Tree Classification Algorithms.* PhD dissertation, The University of Illinois at Urbana-Champaign, 2001.

[65] T. Li, S. Ma, and M. Ogihara. Entropy-based criterion in categorical clustering. In *Proc. of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

[66] Y. Li, L. Wang, and S. Jajodia. Preventing interval-based inference by random data perturbation. In *Proc. of Privacy Enhancing Technologies*, San Francisco, CA, USA, 2002.

[67] Y. Li, S. Zhu, L. Wang, and S. Jajodia. A privacy-enhanced microaggregation method. In *Proc. of 2nd International Symposium on Foundations of Information and Knowledge Systems*, pages 148–159, 2002.

[68] C. K. Liew, U. J. Choi, and C. J. Liew. A data distortion by probability distribution. *ACM Trans. Database Syst.*, 10(3):395–411, 1985.

[69] T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40(3):203–228, 2000.

[70] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proc. of Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference*, pages 36–54, Santa Barbara, California, USA, 2000.

[71] Y. Lindell and B. Pinkas. Privacy preserving data mining. *J. Cryptology*, 15(3):177–206, 2002.

[72] K. Liu, H. Kargupta, and J. Ryan. Random projection-based multiplicative data perturbation for privacy preserving distributed data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):92–106, 2006.

[73] K. Muralidhar, R. Parsa, and R. Sarathy. A general additive data perturbation method for database security. *Management Science*, 45(10):1399–1415, 1999.

[74] K. Muralidhar and R. Sarathy. An enhanced data perturbation approach for small data sets. *Decision Sciences*, 36(3):513–529, 2005.

[75] K. Muralidhar and R. Sarathy. Data shuffling - a new masking approach for numerical data. *Management Science*, forthcoming, 2006.

[76] J. Natwichai, X. Li, and M. E. Orlowska. A reconstruction-based algorithm for classification rules hiding. In G. Dobbie and Eds. J. Bailey, editors, *Proc. of the 17th Australasian Database Conference (ADC2006)*, pages 49–58, Hobart, Australia, 2006.

[77] UCIrvine University of California. Uci machine learning. available from http://www.ics.uci.edu/∼mlearn/MLRepository.html. visited on 12.10.06.

[78] US Department of Labor. Executive order 13145. available from http://www.dol.gov/oasam/regs/statutes/eo13145.htm, February 8 2000. visited on 23.07.06.

[79] The University of Waikato. available from http://www.cs.waikato.ac.nz/ml/weka/. visited on 12.10.06.

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

[81] S. R. M. Oliveira and O. R. Zaïane. Privacy preserving frequent itemset mining. In *Proc. of IEEE ICDM Workshop on Privacy, Security and Data Mining*, page 4354, Maebashi City, Japan, December 2002.

[82] S. R. M. Oliveira and 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'03)*, pages 54–63, Hong Kong, China, July 2003.

[83] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM 2003)*, pages 613–616, Melbourne, Florida, USA, December 19-22 2003.

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

[85] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, California, USA, 1993.

[86] S. P. Reiss. Practical data-swapping: The first steps. *ACM Transactions on Database Systems*, 9(1):20–37, 1984.

[87] Consumer Report. Ibm multi-national privacy survey consumer report. available from http://www1.ibm.com/services/files/privacy_survey_oct991.pdf. visited on 01.07.03.

[88] R. M. Research. Community attitude towards privacy 2004, a survey prepared for the office of the federal privacy commissioner, australia, survey prepared by roy morgan research. available from http://www.privacy.gov.au/publications/rcommunity/index.html, June 18 2004. visited on 23.07.06.

[89] S. Rizvi and J. R. Haritsa. Maintaining data privacy in association rule mining. In *Proc. of the 28th VLDB Conference*, pages 682–693, Hong Kong, China, 2002.

[90] J. Roddick and 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, April 2003.

[91] A. P. Sanil, A. F. Karr, X. Lin, and J. P. Reiter. Privacy preserving regression modelling via distributed computation. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 677–682, Seattle, Washington, USA, August 2004.

[92] R. Sarathy, K. Muralidhar, and R. Parsa. Perturbing nonnormal confidential attributes: The copula approach. *Management Science*, 48(12):1613–1627, 2002.

[93] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, 2001.

[94] Y. Saygin, V. S. Verykios, and A. K. Elmagarmid. Privacy preserving association rule mining. In *RIDE*, pages 151–158, 2002.

[95] E. Schuman. At wal-mart, world's largest retail data warehouse gets even larger. available from http://www.eweek.com/article2/0,1759,1675960,00.asp, 10 2004. visited on 29.03.05.

[96] A. Schuster, R. Wolff, and B. Gilburd. Privacy-preserving association rule mining in large-scale distributed systems. In *Proc. of 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004)*, pages 411–418, Chicago, Illinois, USA, April 19-22 2004.

[97] A. C. Tamhane. Randomized response techniques for multiple sensitive attributes. *The American Statistical Association*, 76(376):916–923, 1981.

[98] H. Taylor. Most people are "privacy pragmatists" who, while concerned about privacy, will sometimes trade it off for other benefits. available from http://www.harrisinteractive.com/harris_poll/index.asp?PID=365, March 19 2003. visited on 23.07.06.

[99] P. Tendick. Optimal noise addition for preserving confidentiality in multivariate data. *Journal of Statistical Planning and Inference*, 27:341–353, 1991.

[100] P. Tendick and N. S. Matloff. A modified random perturbation method for database security. *ACM Trans. Database Syst.*, 19(1):47–63, 1994.

[101] B. M. Thuraisingham. Data mining, national security, privacy and civil liberties. *SIGKDD Explorations*, 4(2):1–5, 2002.

[102] J. F. Traub, Y. Yemini, and H. Wozniakowski. The statistical security of a statistical database. *ACM Trans. Database Syst.*, 9(4):672–679, 1984.

[103] Human Genome Program US Department of Energy. Genomics and its impact on science and society. available from http://www.ornl.gov/sci/techresources/Human_Genome/publicat/primer2001/. visited on 07.07.06.

[104] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 2002.

[105] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, Washington, DC, USA, August 2003.

[106] J. Vaidya and C. Clifton. Privacy preserving naïve bayes classifier for vertically partitioned data. In *Proc. of the Fourth SIAM International Conference on Data Mining*, Lake Buena Vista, Florida, USA, April 2004.

[107] J. Vaidya and C. Clifton. Privacy-preserving outlier detection. In *Proc. of the 4th IEEE International Conference on Data Mining (ICDM 2004)*, pages 233–240, Brighton, UK, November 2004.

[108] A. Veloso, W. Meira Jr., S. Parthasarathy, and M. de Carvalho. Efficient, accurate and privacy-preserving data mining for frequent itemsets in distributed databases. In *Proc. of XVIII Simpósio Brasileiro de Bancos de Dados (SBBD)*, pages 281–292, Manaus, Amazonas, Brasil, 2003.

[109] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, 2004.

[110] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Trans. Knowl. Data Eng.*, 16(4):434–447, 2004.

[111] S. S. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of American Statistical Association*, 60(309):63–69, March 1965.

[112] J. Frand's web page at UCLA page. Data mining: What is data mining? available from www.anderson.ucla.edu/faculty/json.frand/teacher/technologies/palace/datamining.htm. visited on 29.03.05.

[113] Teradata webpage. Wal-mart is making its huge dara warehouse huger. available from www.teradata.com/t/page/ 129223/. visited on 29.03.05.

[114] C. Westphal and T. Blaxton. *Data Mining Solutions: Methods and Tools for Solving Real-World Problems.* John Wiley & Sons, Inc., Toronto,Canada, 1998.

[115] Wikipedia.
Market trends. available from http://en.wikipedia.org/wiki/Market_trends. visited on 07.07.06.

[116] R. L. Wilson and P. A. Rosen. The impact of data perturbation techniques on data mining accuracy. In *Proc. of the 33rd Annual Meeting of the Decision Sciences Institute*, pages 181–185, 2002.

[117] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[118] R. N. Wright and Z. Yang. Privacy-preserving Bayesian network structure computation on distributed heterogeneous data. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–718, Seattle, Washington, USA, August 2004.

[119] A.C. Yao. Protocols for secure computations. In *Proc. of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, 1982.

[120] N. Zhang, S. Wang, and 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*, pages 484–495, Pisa, Italy, September 2004.

[121] Y. Zhu and L. Liu. Optimal randomization for privacy preserving data mining. In *Proc. of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 761–766, Seattle, Washington, USA, August 2004.