# A Robust Data Obfuscation Approach for Privacy Preserving Collaborative Filtering

A Thesis
Presented to
The Academic Faculty

by

**Rupa Parameswaran**

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
Aug, 2006

# A Robust Data Obfuscation Approach for Privacy Preserving Collaborative Filtering

Approved by:


Professor Douglas Blough, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Faramarz Fekri
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Linda Wills
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Sham Navathe
College of Computing
Georgia Institute of Technology

Professor David Schimmel
School of Electrical and Computer Engineering
Georgia Institute of Technology



Date Approved: May 09, 2006

*Dedicated to my loving parents.*

# ACKNOWLEDGEMENTS

First, I would like to thank my advisor Prof. Douglas Blough for his constant encouragement and guidance and especially for inspiring me to pursue a PhD. He provided me with the opportunity to work on several interesting projects during the course of my study. I would like to thank him for involving me in the Data Obfuscation project that led to this dissertation. He has always been there to listen and give advice.

I would like to thank Prof. Linda Wills for her interest in my research. In addition to being a committee member in my proposal and thesis defense, she also took a keen interest in my progress. Prof. Sham Navathe deserves a special mention for his involvement in my thesis. I would like to thank him for his time and insightful advice. I would like to thank the rest of the committee members Prof. David Schimmel and Prof. Faramarz Fekri for their comments, suggestions and willingness to participate in the dissertation defense.

The financial support offered by the department during the last two semesters is highly appreciated. I consider myself lucky to be allowed to instruct a course. I would like to sincerely thank Prof. Paul Steffes for considering me favorably for the position. I would like to thank Marilou Mycko for being there to make sure that I had all my paper-work in order and also for her advice and guidance throughout my course of study. Another person that deserves a special thanks is Suzzette Willingham for her eagerness to help and foe her kind advice.

I would like to thank all my lab-mates over the years. Rahul Motwani, Weilai Yang, and Chris Wood helped me through my first two years at Georgia Tech. I would like to thank Saraswati Bharatipudi for her friendship and assistance through good and difficult times both professionally and personally. I would also like to thank all my friends Sangeeta Bhattacharya, Pradeep Kamath, Harshit Shah, Bala Ganesh, Priya Gopalakrishnan, Vijay Bharadwaj, Vaidya Sankaran, and Jini Khetan for their friendhip and support.

I would like to thank my family for their love and encouragement. Most importantly, I

would like to thank my parents P. V. Parameswaran and T. K. Rukmani for encouraging me to pursue my interests. They have always believed in me and supported me in all my endeavors - thank you.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

Privacy is defined as the freedom from unauthorized intrusion. The availability of personal information through online databases, such as government records, medical records, and voters lists, pose a threat to personal privacy. The concern over individual privacy has led to the development of legal codes for safeguarding privacy in several countries [56]. However, the ignorance of individuals as well as loopholes in the systems, have led to information breaches even in the presence of such rules and regulations. Protection against data privacy requires modification of the data itself. The term *data obfuscation* is used to refer to the class of algorithms that modify the values of the data items without distorting the usefulness of the data. The main goal of this thesis is the development of a data obfuscation technique that provides robust privacy protection with minimal loss in usability of the data.

Although medical and financial services are two of the major areas where information privacy is a concern, privacy breaches are not restricted to these domains. One of the areas where the concern over data privacy is of growing interest is collaborative filtering. Collaborative filtering systems are being widely used in E-commerce applications to provide recommendations to users regarding products that might be of interest to them. The prediction accuracy of these systems is dependent on the size and accuracy of the data provided by users. However, the lack of sufficient guidelines governing the use and distribution of user data raises concerns over individual privacy. Users often provide the minimal information that is required for accessing these E-commerce services. The lack of rules governing the use and distribution of data disallows sharing of data among different communities for collaborative filtering. The goals of this thesis are (a) the definition of a standard for classifying DO techniques, (b) the development of a robust cluster preserving data obfuscation algorithm, and (c) the design and implementation of a privacy-preserving shared collaborative filtering framework using the data obfuscation algorithm.

# CHAPTER I

# INTRODUCTION

## *1.1 Privacy and Collaborative Filtering*

People are faced with choices every day, some more important than others. Most of the time, we seek help from various sources in order to make the right decision. There are several resources from which information can be obtained, such as review boards, peers with similar interests, professional advisers, and information from relevant news articles. Scanning through all the resources can be extremely cumbersome. In some cases, the breadth of information provided by the resources can make the decision process more difficult by increasing the number of available choices. The growth of Internet age has increased the availability of information, further increasing the number of choices.

Collaborative Filtering (CF) systems automate the process of filtering out relevant information that might be of interests to users. These systems identify common patterns or preferences of sets of users and use these preferences to make recommendations regarding articles or items of potential interest to them [60]. Early CF systems required users to seek information from a known set of users. The development of information retrieval techniques led to the implementation of automated CF systems. These systems provide the user with recommendation without the user having to seek information [27]. Other developments in CF systems include the improvement from a completely memory-based approach using nearest-neighbor techniques to a model-based approach using methods like Bayesian clustering. Although several CF systems have been implemented, the improvements in the accuracy of predictions have only been marginal. In order to provide personalized information to a user, the CF system requires sufficient information regarding his or her preferences, behavioral characteristics, as well as demographic information of the individual. The accuracy of the recommendations is dependent largely on how much of this information is known to the CF system. However, this information can prove to be extremely dangerous

if it falls in the wrong hands. The concerns over personal privacy impose a limitation on the amount of information that can be provided to a CF system. Individuals refrain from providing information because of fears of personal safety. The lack of laws governing the use and distribution of this data is one of the prime reasons for these concerns. The accuracy of CF systems is limited by the sparseness of available data. The results of a survey on personal privacy [17] indicates that more than 81% of the people in the survey were willing to provide information as long as their privacy are guaranteed. The implementation of a privacy preserving framework for protecting user information is a step in this direction.

The need for privacy preserving techniques is not restricted to Collaborative filtering systems. All databases that store personal/user information are susceptible to privacy breaches and need to be protected. Databases that store sensitive information such as health records, financial records, government records, and employee information are particularly vulnerable to privacy breaches. Some examples of sensitive databases are given below.

- *Health records*

  When dealing with a patient's medical records, data privacy is of the utmost concern. This requirement fundamentally conflicts with the basic repeatability needs of scientific methods, which require that research results be independently validated and extended. Even in the most controlled circumstances, where the records are stripped of sensitive information such as Social Security numbers and other identifiers, disseminating patient medical records greatly increases the odds of data misuse. In particular, accurate knowledge of a few of the non-sensitive fields in a database leads to the identification of individual records [62]. Judicious medical record obfuscation can preserve data usability for medical research while preserving data anonymity and thus thwarting such attacks. It can work because many medical studies use large collections of confidential patient records, cross-correlating the arithmetic mean of different data items such as blood pressure, cholesterol, time exercising, and the amount of fat in a person's diet. In other cases, small statistical perturbations of each data item are within experimental tolerances.

- *Financial transactions*

  Data protection is also important for databases that contain information regarding billing transactions. The need for privacy in these databases is explained using an example of transponder-based toll collection for toll roads [67]. This business operates using toll booth sensors to track users' individual transponders and bill them accordingly. The privacy of customers using the toll is essential to protect against spying and tracking drivers' habits; records of purchases made by customers are necessary in case of a questionable credit card transaction. Hiding or disguising the sensitive information contained in the billing statements before selling it to a credit card issuer would prevent data from being cross-referenced to driving habits.

- *Military information*

  Military weaponry is another highly sensitive area that could benefit by data protection techniques. Obfuscating statistical information about military-grade weapons capability and technical parameters might make it possible for third parties that do not hold top security clearance, such as academics and consultants, to demonstrate a weapon's effectiveness and feasibility without releasing full accuracy and thus its full military value. This technique would help prevent an adversary from using the distributed information to develop the weapons or countermeasures against them.

These databases are analyzed by applying several different data mining techniques to draw useful inferences from the data. Some of these databases are also required to be available in the public domain, thus increasing their vulnerability to privacy breaches. The sensitive data in these databases require transformation into a form where the individual values are protected without distorting their information content.

Over the past decade, data protection techniques have been proposed for privacy preserving mining of data. The term data obfuscation (DO) is used to refer to data protection techniques that transform the original data into a less sensitive form. DO techniques desensitize the original data by transformations such as the addition of random noise [5], partial suppression [62], swapping [51], and linear transformation [44][45]. In all of these

approaches, the resulting data sets are different from the original data sets and cannot be mapped to their original form. DO techniques perform the transformation in such a way that the aggregates are still preserved in the dataset. Most of the techniques proposed until now cater to specific domains and perform well for a limited set of applications. In the absence of a standard for classifying DO techniques, comparison and performance analysis of the different techniques is not straight-forward. The domain of interest in this research is data mining. Many data mining applications involve learning through cluster analysis. The term *Usability* is used to refer to the usefulness of the transformed data. Usability is measured in this dissertation in terms of the preservation of the inherent clustering of the original data. The need for an obfuscation technique that preserves privacy as well as usability of the transformed data has motivated the design, development, and performance analysis of a robust cluster retaining DO technique in this research.

This thesis is organized as follows: First, some of the existing data obfuscation techniques for data mining applications are discussed. Next, a new data obfuscation approach for privacy preservation of data mining applications, called Nearest Neighbor Data Substitution (NeNDS), is proposed and described in detail. Third, the properties of data obfuscation techniques are examined and measures for comparing data obfuscation techniques are defined. The reversibility metric for measuring privacy and usability metric for measuring the information content preservation are some of the measures proposed in this dissertation. This is followed by an experimental evaluation of NeNDS and a discussion of the strengths and weaknesses of the different data obfuscation techniques. The proposed data obfuscation technique is then applied to a collaborative filtering framework. The concept of collaborative filtering and the types of collaborative filtering systems are discussed. This dissertation extends the CF system to a shared system that can be used by multiple sources. A framework for preserving privacy in CF system is explained and demonstrated with examples. This is followed by an experimental evaluation of the privacy preserving framework by comparing the accuracy of the results using original data with the results using obfuscated data. The dissertation is concluded with a summary of the thesis and proposal of future work in this area.

## *1.2 Contributions*

The goal of this thesis is to develop a privacy preserving framework for shared collaborative filtering. The underlying requirement for privacy is a good data protection technique also known as data obfuscation technique. The steps involved in the development of a privacy preserving framework are (1) defining the meaning of privacy in the context of data mining, (2) assessing the utility requirements for data mining applications, (3) developing a data obfuscation technique for privacy preservation of sensitive data, (4) analyzing the existing collaborative filtering framework, and (5) implementing a model for privacy preserving collaborative filtering.

The main contributions of this thesis are listed as follows:

- A strong attack model for privacy breach: The existing attack models did not consider threats resulting from partial knowledge about the data or the obfuscation process that might be available to the attacker. There are several ways in which the attacker can obtain this information, some of which are listed in Chapter 2.1. The first contribution of this thesis is a strong attack model that covers all scenarios that could lead to a compromise of some or all of the records in the database.

- A robust data obfuscation technique: The goal of data mining applications is to mine database to provide as much accurate information as possible to the end user. This goal seemingly conflicts with data privacy, which attempts to conceal as much information as possible. The contradicting requirements of data usability and data privacy are balanced by understanding the meaning of usability with respect to the applications that use the database. Data mining applications are the domain of interest in this thesis. The main requirement for data mining applications is the preservation of dependencies among data items. The Nearest Neighbor Data Substitution (NeNDS) technique proposed and implemented as part of this research is one of the main contributions in this thesis. NeNDS protects privacy by permuting similar sets of data items. Usability is provided by NeNDS by grouping similar items together before

permutation. The claim of the usability of NeNDS is backed by experimental analysis of cluster preservation for different databases obfuscated by NeNDS. A hybrid of the NeNDS-based obfuscation technique, called GT-NeNDS, is also implemented and evaluated as part of this thesis.

- Privacy preserving properties for data obfuscation: An important part of the research in data privacy involves the definition of "good" privacy. In this context, identifying the characteristics of a data obfuscation technique that makes it a strong privacy preserving mechanism is very important. Data obfuscation properties that make an obfuscation technique robust form an important part of this research.

- Privacy preserving collaborative filtering (CF): The final goal of the thesis is to develop a framework for privacy preserving data mining. The hybrid version of NeNDS, GT-NeNDS, formed the basis of the privacy preserving framework. A NeNDS-based privacy preserving framework also resulted in the proposal of a model for CF in which multiple E-commerce vendors can share their databases with a centralized server. A NeNDS based privacy preserving framework for shared collaborative filtering is the final contribution of this dissertation.

# CHAPTER II

# PROBLEM STATEMENT

The motivation for this thesis is the growing concern over the threat to personal privacy resulting from sensitive databases. The main challenge in overcoming privacy threats is to develop data protection techniques that protect the sensitive data contained in the databases without losing the useful information provided by them. This involves understanding the meaning of privacy and modeling attack scenarios that violate privacy. Privacy preservation requires data to be modified or obfuscated. The requirement that the obfuscated data should still be useful places a bound on the extent to which the data can be modified. It is impossible to achieve data protection without some loss in usability [10]. A good data obfuscation (DO) technique is one that maximizes the usability of the resulting data while protecting privacy of the records in the database. The properties of DO techniques that enable privacy protection need to be identified. A good obfuscation technique is one that exhibits all the properties required for privacy protection. The proposed technique is then validated by evaluating its usability and privacy preservation property. Finally, the applicability of the technique in practical applications needs to be validated.

The problem statement for this dissertation is listed as follows.

- Definition of Privacy: Privacy is defined as the freedom from unauthorized intrusion. A privacy preserving transformation is one which hides the identity of individual records and protects the transformed data from being mapped to their original values. Although several data obfuscation techniques have been developed in the past, no standard has yet been developed to compare the strengths and weaknesses of the techniques. The identification of properties of good DO techniques is the first challenge in this thesis.

- Definition of Usability: The usefulness of the data depends on the applications that use

the obfuscated data. The domain of this research is data mining applications. These applications use different types of clustering techniques for finding useful patterns in the data. The relationship among the data items in the database are analyzed to draw inferences on the characteristics exhibited by different groups of data, the dependencies of the different fields in the database, the anomalies in the database, and so on. The usability requirement for such applications is that the resulting obfuscated database should preserve the dependencies and inter-relationships of the data items.

- The privacy preservation technique: The main goal of this thesis is to propose and implement a robust data obfuscation technique that enables data mining in a privacy preserving framework.

- A privacy preserving collaborative filtering framework: The applicability of the data obfuscation technique is evaluated by applying it to a collaborative filtering framework.

## *2.1 Data Privacy Attack Model*

The first part to developing a data obfuscation technique is to build an attack model to assess the vulnerable points that the attacker can use to compromise the database. There are ways of accessing a database.

1. The user makes queries to a centralized server. In this case, the user does not have access to the entire database, and is provided only with the information requested. The centralized server can store the data in the obfuscated form itself or make queries to the original database and obfuscate the results just before sending it to the user. The first approach is called input-based obfuscation and the latter approach is called response-based obfuscation. The input-based obfuscation approach results in permanently altering the database. This might result in the loss of usable information. The response-based approach returns accurate results, which can be obfuscated depending on the type of user, the kind of query made to the server, and the number of pertinent queries made previously. Although the response-based obfuscation appears to provide

better usability, it is prone to targeted attacks by users. Even though the results are obfuscated, the database is susceptible to attacks where the user adaptively queries the server based on the previous results to compromise the database [10]. The input-based obfuscation approach places a bound on the accuracy of the information that can be provided to the user and prevents the user from obtaining the information that has already been obfuscated.

2. The user is provided with the entire database. In this case, the database is obfuscated to protect all the sensitive data. This has to be done in such a way that preserves the usefulness of the database.

The attack models presented in this chapter assume that the database that the attacker queries are stored in the obfuscated form. This means that the result of the database remains the same every time for the same query posed by the attacker. The models presented here are applicable for both data access scenarios. The discussion also assumes a query-based centralized database. However, the models are also applicable when the attacker has access to the entire obfuscated database.

### 2.1.1 Existing Attack Model

The attack model for DO is different from the attack model for encryption-based security techniques, but no common standard has been implemented as yet for the analysis of DO techniques. Each of the proposed DO techniques uses a different form of comparison of the effectiveness of the approach. Existing work on the privacy analysis of DO techniques has primarily considered a model where the attacker correlates obfuscated responses with data from other publicly-accessible databases in order to reveal the sensitive information of interest. The attack model is shown in Figure 1.

The attacker is assumed to have access to publicly available databases, some of which are unobfuscated. In this attack model, the attacker makes queries to the target database and compares the results of the query with the records in the databases that he has access to. The privacy preserving techniques proposed so far assume this type of attack model. In this model, a privacy breach results when an attacker is able to obtain previously unknown

information about one or more records from the obfuscated database by comparing and correlating the database contents with other databases that he is able to access. Although this attack model is a likely scenario for compromise, it does not model all the modes of attack that lead to a breach of privacy.

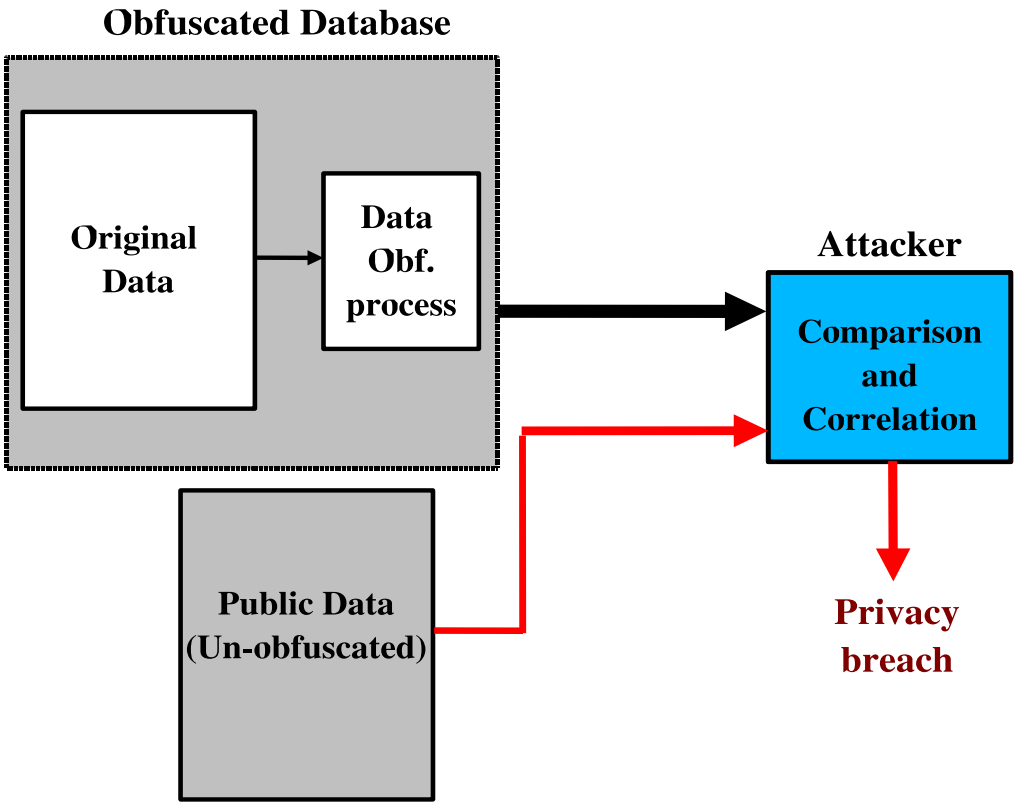



**Figure 1:** Existing attack model for analysis.

### 2.1.2 Proposed Attack Model

The model proposed in this dissertation is shown in Figure 2. The previous model assumes that the attacker has no *a priori* knowledge of the process used for data obfuscation. It also assumes that the attacker does not know any of the entries in the target database. These are impractical assumptions. There are several reasons for assuming that the attacker may have access to some of the records in the database. Some of the reasons are listed below:

- The attacker is a part of the database. In this case, he has information regarding at least one record in the database.
- The attacker has a friends or family members who are a part of the database. In this case, the attacker has knowledge of a finite number of records in the database.
- The attacker was involved in collecting a part of the information in the database. In this case, the attacker has complete knowledge of that part of the database.
- The attacker is the administrator of the database. In this case, the attacker has complete knowledge of the obfuscation process used for data obfuscation.

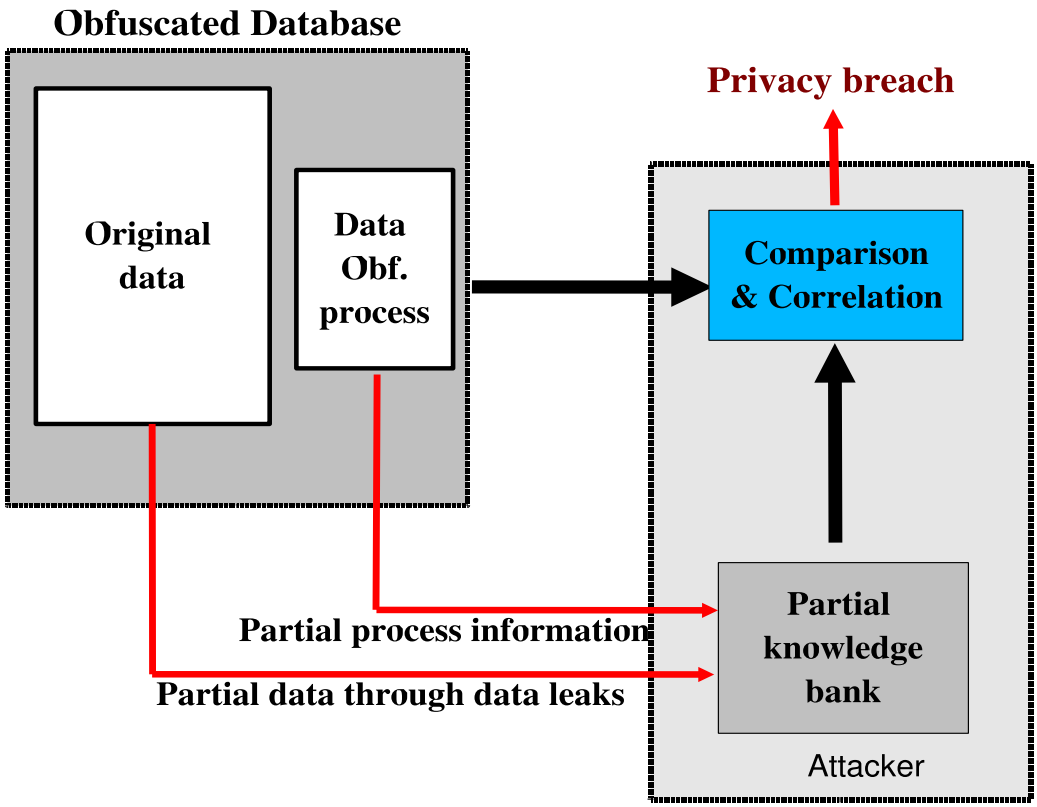



**Figure 2:** Proposed attack Model for analysis.

These are only a few ways in which an attacker can gain access to some information about the database. Hence, the attack model should include the partial knowledge available

to the attacker. The attack model in Figure 2 includes side channels from the original target database and the data obfuscation process to model the partial information gained by the attacker. The attacker can then use this partial information to attempt to reverse engineer the entire data set.

One useful byproduct of this model is a measure of the robustness of a data obfuscation technique, namely the percentage of the unobfuscated data set that an attacker must know in order to be able to learn the entire set. This new measure is used to evaluate the robustness of existing DO techniques. It has been demonstrated in this thesis that many well-known data obfuscation techniques are highly vulnerable to reverse engineering through unintentional release of only a small percentage of the unobfuscated data set. In this attack model, the privacy provided by a DO technique can be measured by the amount of information required for reverse engineering the obfuscated data.

## *2.2 System Model*

Two system models are used in this dissertation, a generic data mining system model, and a collaborative filtering system model. The generic system model evaluates the privacy and usability of databases obfuscated using data obfuscation techniques for clustering-based data mining. Privacy is defined for this model as the protection of individual records in the obfuscated database from being identified or being mapped back to the original records. The difficulty of reverse-engineering the original record(s) from the obfuscated database is a measure of the privacy provided by the data obfuscation technique. The usability of the obfuscated database in the generic system model is measured by evaluating the distortions introduced in the clusters. Clustering algorithms are applied to the original and obfuscated databases, and the distortions in the clusters formed by the obfuscated databases are measured. The extent of the distortion in clustering is a measure of the usability in the generic system model.

The collaborative filtering system model evaluates the applicability of the obfuscation technique in a practical application, which in this case is collaborative filtering (CF). A CF system uses data mining techniques to identify the purchase patterns of users by using the

demographic information of the users, their purchase histories, and the ratings/feedback provided by the users on the purchased items. The CF system uses the purchase patterns to predict the items that users might be interested in purchasing and recommends the predicted items to the users. The input to this system is a set of three databases, the first database containing demographic information regarding the user, the second database containing the items in the store's inventory, and the third database containing the ratings provided by users on the purchased items. The output of this model is the database containing the recommended ratings for items that have not been purchased by users. The attack model only considers the user information and the ratings information databases as sensitive databases. Hence, only these databases are obfuscated before applying collaborative filtering techniques to mine the databases. In this model, privacy is defined as the protection of individual records of the user database as well as the database containing the ratings provided by users. Usability is measured as the decrease in accuracy of the predictions as a result of data obfuscation.

The generic system model evaluates the cluster-preserving property of DO techniques by applying clustering algorithms and measuring the distortions in the resulting clusters. The collaborative filtering model uses clustering, similarity, probabilistic techniques on the database to make recommendations. This model evaluates the extent to which the results are distorted when the results of data mining are used to make predictions.

# CHAPTER III

# BACKGROUND AND RELATED WORK

This chapter is divided into two sections. The development of collaborative filtering techniques and research related to privacy in collaborative filtering is described in Section 3.1. The next section discusses privacy preservation of databases in general. The different data protection techniques and their limitations are discussed in Section 3.2. In this thesis a new data obfuscation technique for data mining applications is proposed, developed and implemented in a collaborative filtering system.

## *3.1 Collaborative Filtering*

The term 'Collaborative Filtering' (CF) was first introduced in the Tapestry system [21], for filtering electronic documents through e-mail and Usenet postings. In this system, a user explicitly requests recommendations based on reviews of a specific set of known individuals. The drawback of this system is that it requires a close-knit group of people who are aware of each other's interests. The lack of scalability of this system for larger networks led to the development of more Automated Collaborative Filtering systems (ACF) [54]. The GroupLens CF system [52] pioneered the research on ACF by using pseudonymous users to provide ratings for movies and Usenet news articles. Some of the other recommendation systems such as the e-mail based music recommendation system [66], Ringo, and the web-based movie recommendation [30], Video Recommender, also developed ACF algorithms for recommendations. All three systems use neighborhood-based prediction algorithms such as Pearson's correlation and vector similarity. These algorithms are referred to as memory-based algorithms because they use the raw data in the database to make recommendations. Model-based approaches such as Bayesian network models and cluster-based models were proposed in [65][16]. These algorithms first develop cluster-based models or Bayesian network models on the database. The models are then used for making predictions for users

on items that have not yet been rated by them. This makes model-based CF algorithms faster and less memory-intensive. Hybrid memory-model based approaches have also been developed to improve accuracy of predictions [49].

As with any system that stores personal information of individuals, CF systems are vulnerable to privacy invasion. Although meta-store fronts such as *Amazon*, *C-net*, *Yahoo* assert privacy policies that protect user data, their policies are intentionally vague in certain areas. For instance, Amazon's policy states that in the event that the company is bought over, the personal assets are subject to be transferred to the parent company. Such loop holes in the policies present privacy concerns resulting in users refraining from divulging any personally identifiable information. This results in incomplete or sparse databases. The absence of complete information or dense databases affects the accuracy of the recommendation systems. Privacy preservation by factor analysis [12][14] proposes a secure computation technique using homomorphic encryptions. Here users' ratings are stored as encrypted vectors and aggregates of the data are provided in the public domain. This approach requires the users to seek out recommendations explicitly. The random perturbation approach proposed in [50] uses a noise vector to mask the original data. Although the technique permits heterogeneous diffusion based recommendations, the accuracy of the predictions is dependent on the amount of noise added. The drawbacks of random perturbations are discussed in [46]. In this paper, we propose a frame-work for privacy preserving collaborative filtering by a hybrid NeNDS based data obfuscation approach.

Secure recommendations using trust-based CF techniques have been proposed in [37][38] to protect against targeted attacks to push a chosen set of items. Such attacks, known as shilling attacks [69][35] are achieved by introducing false profiles in the database that rate a chosen set of items in such a way that their overall rating changes significantly. Trust-based systems prevent such attacks by introducing a web of trusted users whose ratings are preferred over the un-trusted users. While trust-based systems protect the truthfulness of the ratings and avoid attacks on the CF system, the privacy framework attempts to protect the personally identifiable fields of individuals participating in the ratings. A secure CF system should protect the quality of the recommendations as well as the privacy of the

participants that provide the ratings.

## *3.2 Data Obfuscation Techniques*

The abundance of information available online has resulted in the loss of individual privacy [18]. Several methods have been proposed and implemented for privacy preservation of sensitive data sets [32]. The term *data obfuscation* [7] is used as a generalization of all approaches that involve distorting the data for privacy preservation and other purposes. One of the more common techniques is cryptography, where sensitive data is encrypted with a key and is accessible only to an authenticated user. In several applications, it is necessary to provide different levels of precision of data, based on the type of user requesting access. The encryption of data does not provide this capability. The usability of the data is therefore restricted only to a narrow set of users. Secure multi-party encryption techniques propose to perform computations on data in the encrypted form [55].

Techniques have been developed to perform order-preserving encryption on data [4]. These techniques, however, are limited only to simple addition and multiplication operations and to date have not been applied to more complex statistical computations. For trend analysis and statistical and inference-based computations from data sets, encryption-based security schemes add complexity without much benefit in terms of privacy.

Privacy laws such as the HIPAA laws for protecting medical records, and the Gramm Bleach and Bliley act for financial record privacy use access control based methods for data protection. Access control policies for databases, such as Hierarchical access control policies [9] and role-based access control policies [59] have been proposed. These methods protect the data from unauthorized users. Policy-based protection does not preclude misuse of the data by authorized users either intentionally or unintentionally. Access control policies arenecessary but not sufficient to provide robust privacy protection. Data obfuscation techniques attempt to protect the data by applying transformations that retain the usefulness of the data but protect the sensitive information. In this case, authorized users are not required to be provided with the original data, thereby providing better privacy protection.

Privacy preservation by data randomization is based on adding a noise vector to the original data, thereby desensitizing the precise information content [5]. Data randomization mainly operates on a subset of database tables, fields, and records and is designed to maintain the statistical properties of a database. Unless the noise distribution follows the distribution of the original data, information regarding dependencies among the attributes would be lost.

Data anonymization [34] attempts to classify data into fixed or variable intervals. The usefulness of the obfuscated data and the privacy factor are dependent on the choice of the interval. A large interval makes the data less useful, while an interval that is too small does not provide sufficient privacy protection of the data. In [62], the author proposes a generalization and suppression approach to obtaining the required anonymity level: generalization replaces a value with a less specific value, while suppression does not release a value at all.

This guarantees that each data item will relate to at least $k$ other entries, even if the records are directly linked to external information. K-anonymization has been proven to be an NP-hard problem [39]. Various algorithms, such as the k-optimal anonymization algorithm [62], the simulated annealing technique [68], and the condensation-based k-anonymization [2], have been proposed to optimize and solve the generalization/suppression problem, but even the most optimum algorithm that uses an approximation technique now has a polynomial complexity. The other drawback of the anonymization technique is the loss of information. The generalization approach categorizes quantitative information into intervals, thus reducing the granularity of the information. Furthermore, data entries that are not possible to generalize are suppressed. This leads to a complete loss of information regarding certain fields.

The term *usability* pertains to the usefulness of the data that has been obfuscated. The most important characteristics that must be preserved for data mining applications are the multivariate statistical distributions as well as the clustering property of the data. An optimum data obfuscation technique is one that preserves both these properties while still providing strong privacy preservation. One of the techniques that proposes to preserve usability while preserving privacy is Geometric Transformation [44] [45]. In this approach,

geometric transformations such as rotation, scaling, and translation are used to obfuscate the data. This type of obfuscation is proposed for preserving the inherent clustering information of data. As geometric transformations are isometric, the transformed data retains its isometric properties. While this technique does involve "modifying" data, the interrelation of the data elements within the data sets and across the fields is maintained even after the obfuscation. This type of approach is useful in applications where the data needs to be disguised completely, such as the third-party mining of sensitive data. Geometric transformation-based obfuscation is weak in terms of privacy preservation and is unsuitable for use in sensitive databases. Random data perturbation, as well as anonymization, result in the "modification" of data. This results in slight differences of the characteristics between the original and obfuscated distributions. Such differences are likely to be very small for large data sets, but are observed to be significant for smaller data sets [42]. Owners of sensitive databases, such as the Census Bureau, look unfavorably upon such "modifying" data obfuscation techniques. Since the preservation of multi-variate characteristics while preserving privacy is an intractable problem, the next important statistical characteristic to preserve is the marginal distribution characteristics. One of the obfuscation techniques that have been widely adopted for sensitive data protection is data swapping [23]. The concept of obfuscating data sets by swapping the elements in the data set was first proposed in [51]. This technique intelligently swaps entries within a single field in a set of records so that the individual record entries are unmatched, but the statistics are maintained across the individual fields. Swapping can be implemented such that the swapped values are close to each other, thus approximating the information in the non-obfuscated data records [19]. As data swapping does not "modify" the actual values of the data, the characteristics of the marginal distributions of the variables are preserved exactly [42].

The requirement of preserving privacy as well as the usability of sensitive data has led to the proposal and development of a robust data obfuscation technique called *Nearest Neighbor Data Substitution (NeNDS)*. The underling principle of this technique is a more generalized version of data swapping. In NeNDS, sets of data that are close to each other

in Euclidean space are grouped together into neighborhoods. The data within each neighborhood are permuted in such a way that the original values are replaced by one of the neighbors in a non-reflective manner. The non-reflective condition is enforced to avoid the swapping of data and to make it less vulnerable to reversal. This approach benefits from the advantages of data swapping, and proposes a more robust privacy-preserving scheme for data obfuscation.

A hybrid version of the NeNDS transformation technique is also proposed here for applications where the usability lies only in cluster preservation and the data is required to be completely disguised. The weak privacy preservation of geometric transformation makes it unsuitable for use by itself. The proposed technique, called GT-NeNDS, uses geometric transformations to mask the data and performs a data substitution on the transformed data. Since GT-NeNDS is a combination of two DO techniques (NeNDS and geometric transformations) that preserve the inherent clusters in the data, GT-NeNDS also exhibits the cluster-preserving property. The robust privacy protection provided by NeNDS strengthens the transformation process and results in a robust data-disguising obfuscation technique for data mining applications.

The analysis of the data obfuscation techniques requires a standard for comparison. No common standard has been implemented as yet, and each of the proposed obfuscation techniques uses a different form of comparing the effectiveness of the approach. The classification of data obfuscation techniques based on their ease of reversal is proposed here. The property, called *reversibility*, is described and categorized for different types of transformations.

# CHAPTER IV

# DATA OBFUSCATION PROPERTIES

The term data privacy is broadly defined as the implementation of appropriate safeguards to ensure the security and confidentiality of data records. To implement appropriate safeguards, it is necessary to first understand the nature of the application using the data as well as the security and confidentiality threats that need to be protected against. Hiding too much of information results in loss of data usability, while insufficient protection poses a threat to data privacy. Although several Data Obfuscation (DO) techniques have been proposed for the protection of data privacy, no standard has been developed as yet for measuring or comparing DO techniques. This chapter identifies the different aspects of privacy protection and proposes *Data Usability* and *Data Privacy* as properties that can be used to evaluate the strength of DO techniques. The *Data Usability* property defines the usefulness of the data obfuscated by a DO technique, while the *Data Privacy* metric measures the extent of privacy provided by the DO technique.

## *4.1 Data Usability*

The main difficulty in finding a solution to the problem of privacy is the contradicting requirement of hiding sensitive data while still providing useful information content. The definition of "useful information content" varies from one database application to another. Similarly, the definition of "sensitive information" is specific to each database application. The problem of privacy is formally defined in this research as the protection of sensitive information of individual records or a subset of sensitive records while preserving the aggregate information contained in the database. A good DO technique is one that hides information about individual records in the database, but provides accurate information in the aggregate. The term *Data Usability* refers to the ability of a DO technique to provide accurate aggregate information. In this research, two aspects of *Data Usability* are

considered.

- Preservation of statistical information: One of the most common applications of databases is to draw statistical inferences from the data. The moments, marginal statistics, and multivariate statistics of the obfuscated data are analyzed and compared with those of the original data. The distortion in the statistical results resulting from the DO technique provides an estimate of the *Data Usability* of the DO technique for statistical purposes.

- Preservation of clusters: Clustering-based techniques are extensively used in data mining applications to study interesting patterns in databases. Obfuscated data that loses its original clustering becomes unsuitable for data mining purposes. Hence, cluster preservation is a critical factor in selecting a DO technique for such databases. In this research, the distortion in the clusters resulting from the obfuscated database provides an estimate of the *Data Usability* of the DO technique for data mining purposes.

An ideal DO technique is one that preserves both statistical information as well as clustering information. Data randomization techniques [5], which obfuscate data by the addition of random noise to the original data, can be tailored to preserve statistical information. However, the inherent clusters in the original data are distorted because of the addition of random noise. Data anonymization techniques [62], which categorize sets of $k$ similar records by a process of suppression and generalization, can preserve statistical information for small values of $k$, but fail to preserve the original clusters because of the process of suppression and generalization. Geometric transformation techniques [44] obfuscate the data using linear transformations such as rotation, scaling, and translation. These transformations distort the statistical distributions of the data. Geometric transformation techniques preserve the original clusters by virtue of their linearity property and are suitable for data mining applications. Data swapping, which obfuscates by a process of swapping nearest neighbors, preserves statistical moments over individual datasets. NeNDS, which is the data obfuscation approach proposed in this research, obfuscates data by permuting amongst similar data items. *NeNDS* preserves all statistical moments over each dataset in

the database, but fail to preserve multi-variate statistics. *NeNDS* also preserves the original clusters even after obfuscation. The cluster preservation property of *NeNDS* is evaluated experimentally in Section 6.2.

## *4.2 Data Privacy*

The definition of privacy is dependent on the type of data that needs to be protected as well as the target applications that use the data. In some privacy sensitive databases or applications, distortion of the original data such that it is similar but not exactly identical to the original data is considered as *acceptable privacy*. In other applications, any similarity between the obfuscated data and the original data is *unacceptable* and is equivalent to an invasion of privacy. Similarly, in applications such as medical databases, the breach of even a single record is *unacceptable*. However, in the case of publicly available databases, the breach of up to *two percent* of the total database is considered as *acceptable* privacy. Hence, any method that measures the strength of DO techniques needs to address the different requirements of privacy for different applications. The privacy measure proposed in this research measures the strength of DO techniques based on three aspects of privacy invasion.

- Approximate privacy invasion: This refers to the ability of a DO technique to conceal any similarity between the obfuscated data and the original data.
- Absolute privacy invasion: This refers to the ease or difficulty of retrieving the original data accurately from the obfuscated data.
- Susceptibility to partial privacy invasion: This refers to the ease or difficulty of retrieving a part of the original data from the obfuscated data.

Privacy against approximate invasion, absolute invasion, and partial invasion cover the basic requirements of privacy for any sensitive database or application that uses the sensitive data. The property of *reversibility* is proposed here to characterize the privacy provided by DO techniques with respect to the three aspects of privacy invasion. The term *reversibility* is defined as the property that dictates the ease or difficulty of the process of reverse engineering obfuscated data [7]. The reversibility property provides a measure of the robustness

to privacy protection that is provided by a DO technique. The reversibility property exhibited by a DO technique can be measured either by the *time* required for reverse engineering the data or by the amount of *a priori* information required to reverse engineer the rest of the original data from the obfuscated data. In this research, the amount of *a priori* information that leads to a privacy breach is used as a measure of *reversibility* of DO techniques.

## *4.3 Reversibility - A Standard for Classification*

Cryptanalysis is used for analyzing the security provided by encryption-based techniques [61]. Since encryption is a deterministic and reversible process, cryptanalysis assumes the transformation to be deterministic as well as reversible. However, DO techniques have no such restriction and therefore require a new standard for analysis.

This section provides a hierarchical classification of the property of *reversibility* as shown in Figure 3. The reversibility property is divided into two main categories, *process reversibility* and *irreversibility*, based on whether the DO technique can be reverse engineered to retrieve the original data. Each of these categories is further classified based on the different attributes of *reversibility* exhibited by various obfuscation techniques.

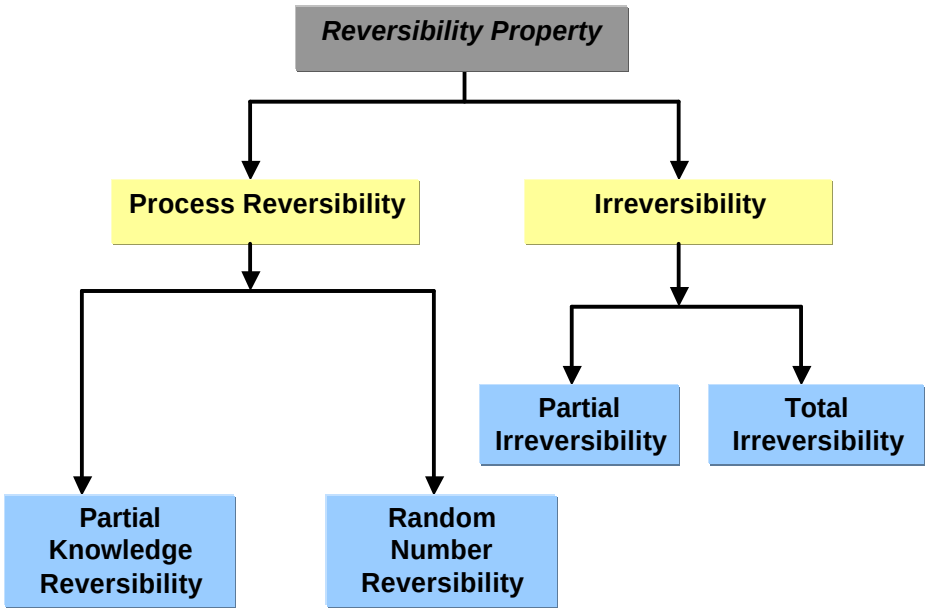



**Figure 3:** Classification of the reversibility property.

An obfuscation technique that can be reversed with the knowledge of the process is

known as a process reversible transformation function. A model similar to the cryptanalytic attack model may be used for this category of transformations. Cryptanalysis of encryption techniques proves the weakness of algorithms to one or more of the well known attack models: plain-text attacks, chosen-plain-text attacks, and chosen-cipher-text attacks. Similarly, process reversible DO techniques can be analyzed with respect to their vulnerability to complete reversal under one or more of the following conditions: with no *a priori* information, with some *a priori* information of the DO process, and with complete *a priori* knowledge of the DO process. *Process reversibility* is sub-classified into the following categories.

1. Partial knowledge reversibility: *Partial knowledge reversibility* implies that a transformation function exhibiting this property can be reverse engineered with the knowledge of either some of the original data entries or a combination of some original entries of data and some information regarding the process used. The level of difficulty of the reversal process is dependent on the DO technique. Obfuscation techniques that involve a *one-to-one* mapping between the original and the transformed data, are vulnerable to *partial knowledge reversibility*. The *reversibility* analysis for linear and non-linear one-to-one transformations is provided in Section 6.1.3.

2. Random number reversibility: This property indicates that the original data set can be reverse engineered with knowledge of the process, the *Pseudo-Random-Number Generator (PRNG)*, and the seed. Most obfuscation techniques invoke PRNGs to generate random sequences. The robustness of DO techniques exhibiting this property relies in protecting the PRNG sequence. As long as the random seed and the sequence are unknown to the attacker, the obfuscated data is robust to reversal. Once this information is revealed and the obfuscation process is known, the entire data is compromised. Transformations that fall under this category cannot be analyzed using cryptanalysis due to their non-deterministic nature.

Obfuscation techniques that result in a non-invertible transformation exhibit *irreversibility*. A *Maximum-likelihood reversibility* estimate can be made in the case of some of the

techniques, which provides an estimate of the confidence with which a guess can be made on the original data. Cryptanalysis fails to account for such transformations as well. With irreversible techniques, there is an inherent loss of information. Lossy compression techniques and data generalization techniques, which make it impossible to exactly recover the original data, fall under this category. The second category of *irreversibility* contains the set of obfuscation techniques in which a part of the obfuscated data becomes irreversible during the transformation. An example of *partial irreversibility* is substitution with repetition, where each data element is replaced by its nearest neighbor. Data elements that are not nearest neighbors of any other element are not included in the final data set and are lost completely. In such cases, the elements that are eliminated from the database cannot be exactly restored by any reversal process.

### 4.3.1 Classification of Transformations

Reverse engineering can be performed with or without *a priori* knowledge of partial information regarding the data obfuscation technique. If no prior information regarding the process is available to the user, the reversal is a probabilistic estimate where the confidence in the estimated process increases with the number of records available. The confidence in the estimated process converges with every additional record for which the conditions of the process are satisfied. The confidence remains a fraction until the process is verified for every record in the database. Thus, in the absence of any prior knowledge regarding the process used in data obfuscation, reversibility can be measured only in terms of a probability, with the confidence in the estimated guess increasing with the number of original records in the database.

In the second case, where the attacker has *a priori* knowledge of a subset of possible processes, the complexity of the reversal process is related to the complexity of the reversal of the processes in the subset. In the simplest case, the guess estimate made by the user based on the set of available records fits exactly one of the processes in the subset of possible processes. This can be estimated either by a set of records that uniquely identify the process or by a set of records cannot identify the rest of the processes in the subset of

likely contenders. In the first case, the number of records required to uniquely determine this process is the measure of the reversibility. For invertible processes, the upper bound on the number of records needed is $m - 1$, where $m$ is the total number of records in the database. In the second case, we study the complexity when more than one process satisfies the guess made by the attacker. In this case, the attacker would require the knowledge of the particular records that distinguish it among the contenders.

Data transformation techniques are classified into three categories and are analyzed with respect to the ease of reverse engineering the obfuscated data.

1. Linear bijective transformation: A function $f : X \rightarrow Y$ is said to be bijective if and only if there is exactly one $x \in X$ with $y = f(x),\ \forall y \in Y$. This implies that there exists a unique function $f(x)$ that maps every element $x$ in the domain $X$ to a unique element $y$ in the co-domain $Y$. Linear bijection describes the set of bijective transformation functions that satisfies the conditions of linear functions. Geometric transformations such as rotation, translation, and scaling are examples of linear bijective transformations. Another example of a linear bijective transformation function is $f(x) = Cx$, where $C$ is a non-zero constant. Here, every element in the co-domain $Y$ is mapped to a unique element in the domain $X$. These transformation functions exhibit the property of partial reversibility and provide weak privacy preservation. The reversibility of linear bijective transformations is further evaluated in Section 6.1.2.

2. Non-linear bijective transformation: A non-linear bijection is a bijective transformation function that has an uncertainty factor added to the transformation. The functions that fall under this category do not fulfill the linearity properties of composition and scaling. An example of a non-linear bijection function is given as $f(x) = x^3 - 1$. Here, the non-linearity lies in the fact that $f(x)$ is a polynomial function, but the bijective property still makes it process reversible. Non-linear bijections can be reversed with knowledge of either the process involved or a minimum of $k$ records from the original database, where $k$ is dependent on the type of function used. The proposed data obfuscation technique, *NeNDS*, is an example of this type of transformation.

Although it exhibits partial reversibility, the fraction of the original data that is required for reversal is large. The fraction of the number of original elements required for reversal of *NeNDS* is evaluated in the next section.

3. Surjective (but not bijective) transformation: A function $f : X \rightarrow Y$ is surjective if, $\forall y \in Y$, $\exists x \in X$ with $f(x) = y$. A surjective (but not bijective) transformation function implies that multiple elements in the domain map to a single element in the co-domain. From the reversibility perspective, such a transformation function is hard to estimate since some elements in the obfuscated data reflect back to more than one entry in the original database. As a result, the attacker would retrieve the $k$ entries in $X$ that map to the entry $y \in Y$, but in the absence of knowledge of the original value, there is no way to distinguish the correct pre-image.

A DO technique that results in a transformation that is difficult to reverse-engineer is robust to partial privacy invasion. The nonresemblance property makes a DO technique robust to approximate privacy invasion. A robust privacy preserving DO technique is one which results in a transformation that is difficult of reverse-engineer and that bears no nonresemblance to the original data.

1. Difficult to reverse-engineer: This property is measured based on the time required to reverse-engineer and the amount of information required to reverse-engineer the obfuscated data. An irreversible transformation is regarded as requiring infinite time and infinite amount of information for reverse-engineering.

2. Nonresemblance: A good data obfuscation technique is one that transforms the data into a form where the individual data items are significantly different from the original data items.

The usability property is measured based on the ability of the data obfuscation technique to transform the data into a privacy preserving form without distorting the inter-relationship of the data items. The distortions in the inherent clustering in the original

data are a measure of the usability. DO techniques are evaluated based on these properties in Chapter 6.1.

# CHAPTER V

# NEAREST NEIGHBOR DATA SUBSTITUTION

## *5.1 Existing Data Obfuscation Techniques*

The need for privacy preserving techniques for protecting sensitive data has resulted in the development of different data obfuscation (DO) techniques. Some of the existing data obfuscation techniques are described in this section.

### 5.1.1 Data Randomization

Data Randomization is one of the first data obfuscation techniques proposed for privacy preserving data mining. In this approach, a data set is obfuscated by adding a random value 'r' to each item in the data set, where 'r' is drawn from a uniform distribution $[-\alpha, \alpha]$ or a Gaussian distribution $[\mu, \sigma]$.

The data is stored in the randomized form along with the parameters for the noise distribution. The parameters are needed to reconstruct the original distribution before the data can be used. The reconstruction process involves the generating a second distribution with the stored distribution parameters, which is then subtracted from the randomized data set. The resulting database preserves the distribution of the original data set although the actual values of the data items have changed. The reconstructed distributions are accurate if the noise distribution function is similar to the distribution function of the original data. For instance, if the original data is distributed follows a normal Gaussian distribution and a Gaussian noise distribution is added to it, the reconstructed data preserves the distribution exactly. However, it has been shown in [5] that the there is a loss of accuracy $(0 - 10\%)$ if the distributions do not match.

The randomized database stores the distributions of the individual data sets, but the inter-relationship of the data across the different fields of the database are not preserved in this type of obfuscation.

### 5.1.2 Data Anonymization

Data anonymization (k-anonymity)uses a generalization/suppression approach for data obfuscation. Generalization is the process of making the data less precise but still consistent. Replacing the birthday of an individual by just the birth year is an example of data generalization. Suppression involves the removal of a data item from the database. K-anonymity iteratively suppresses and generalizes data items in the database until each record is identical to 'k' other records in the database.

### 5.1.3 Data Swapping

Data swapping involves ranking data items in a pre-specified order and swapping records that are close to each other. In contrast with Data randomization and K-anonymity, Data swapping does not "modify" the values of the database. The data are moved around without altering the values.

### 5.1.4 Geometric Transformations

An overview of the geometric transformation-based data obfuscation proposed in [44] [45] is given here. This technique makes use of transformations such as rotation, scaling, and translation for distorting the data [24]. The advantage of using geometric transformations is that these transforms are cluster preserving. Rotation, translation, and scaling are isotropic transformations and hence preserve the distances between the data points even after transformation. Isotropy and similarity are properties of geometric transformations that retain the shape of an image even after transformation. This ensures the usability of the database for clustering applications, where the relationships of the data points are analyzed. An analysis of geometric transformations is presented in this paper with respect to their privacy-preservation property.

Basic transforms: The following section describes the basic geometric transformations in a two-dimensional space. Here, any pair of numerical fields in the database is visualized as an image and the coordinates of the data items are distorted by geometric transformation. The approaches can also be scaled to three or more dimensions without loss of generality.

The database is represented as a two-dimensional space $D_{k,n}$, where $k$ is the number of attributes and $n$ represents the number of instances in the database. The transformations translation, scaling, and rotation can be implemented using matrix multiplication. A general affine transformation is represented by Equation (1). Each of the three transformations can be represented in terms of this equation. Translation is obtained by defining just the $B$ matrix and reducing $A$ to an identity matrix, as shown in Equation (2). Defining only matrix $A$ as shown in Equation (3), results in a pure scaling transformation. For a pure rotation, the matrix $B$ is reduced to 0, and $A$ is defined as shown in Equation 4. The transform matrices for these transformations are given next.

$$\begin{bmatrix} X\prime \\ Y\prime \end{bmatrix} = A* \begin{bmatrix} X \\ Y \end{bmatrix} + B \tag{1}$$

In all of the transformations, $A, B$ are the transformation matrices, $(X, Y)$ are the original data, and $(X\prime, Y\prime)$ are the results of the transformations on the original data.

$$T: \quad A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = + \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} \tag{2}$$

$$S: \quad A = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3}$$

$$R: \quad A = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4}$$

The matrices in (2), (3), (4) represent the matrices for translation, scaling, and rotation respectively. The rotational transformation is more complicated than scaling and translation, as the distortions along the coordinates are inter-dependent. From the description of the transformations, it can be observed that each data set is distorted by the same amount relative to the placement of the individual elements in the set. In this way the clusters are maintained during obfuscation. In the following sections, the robustness of this technique with respect to privacy preservation is analyzed in greater depth.

## 5.2 Nearest Neighbor Data Substitution

The Nearest Neighbor Data Substitution(*NeNDS*) technique is a lossless data obfuscation technique that preserves the privacy of individual data elements by substituting them with one of their Euclidean space neighbors. NeNDS uses a permutation-based approach in which groups of similar items undergo permutation. The permutation approach hides the original value of a data item by substituting it with another data item that is similar to it but not the same. NeNDS treats a database $\Sigma = [\Sigma_1, \Sigma_2, \cdots, \Sigma_m]$ as $m$ individual data sets. Each data set represents one field or attribute of the database such as the *Age*, *Income*, and *Zipcode*. The substitution process is applied to each data set individually independent of the rest of the data sets. The individually substituted data sets $\Sigma_i{}', i \in [i, m]$ are then recombined to form the obfuscated database $\Sigma' = [\Sigma_1{}', \Sigma_2{}', \cdots, \Sigma_m{}']$. The first step in NeNDS is the creation of similar sets of items called neighborhoods. Each data set is divided into a pre-specified number of neighborhoods. The items in each neighborhood are then permuted in such a way that each item is displaced from its original position, no two items undergo swapping, and the difference between the values of the original and the obfuscated items is minimal. The minimum number of neighbors in a neighborhood is specified by the neighborhood size parameter $NH_{size}$, where $[3 < NH_{size} < N]$, and $N$ is the size of the data set. If $NH_{size} = 1$, each item in the data set forms a neighborhood. Permutation cannot take place since there is only one data item in a neighborhood. For $NH_{size} = 2$, the data set is divided into $N/2$ neighborhoods, each containing 2 neighbors. In this case, the two data items can only be swapped with each other, which limit the level of obfuscation possible. Hence, the minimum size of a neighborhood for data obfuscation using NeNDS is $NH_{size} = 3$. The way in which the data items in a data set are permuted differs with different values for $NH_{size}$ resulting in a new obfuscated data set for each value of $NH_{size}$. The number of neighborhoods $NH$ that are created for a data set is given as $NH = \frac{N}{NH_{size}}$. The choice of value for $NH_{size}$ is made by the owner of the database based on the required level of privacy. A larger value of $NH_{size}$ implies that the number of items that are permuted together is large, which leads to better privacy protection. The level of privacy provided by NeNDS for different sizes of neighborhoods is discussed in Section 6.1.3.

The substitution process is performed by determining the optimal permutation set subject to the following conditions.

1. No two elements in the neighborhood undergo swapping.

2. The elements are displaced from their original position.

3. Substitution is not performed between duplicate elements.

These three conditions ensure that each element in the transformed dataset is different from the original dataset. The reflective nature of data swapping makes it vulnerable to partial reversibility. Consider an example where an attacker has information regarding one of the records $X_i = [x_1, x_2, \cdots, x_m]$ in the original database. Let $Y_i = [x_1, z_2, \cdots, y_m]$ be the new record with which the data item $x_1$ has been swapped. Since the attacker knows that $x_1$ is the true value of $X_i$ and that $x_1$ has been swapped with $y_1$, he now knows that the true value of the first field of $Y_i$ is $y_i$. This shortcoming is overcome by avoiding swapping of data items. The displacement condition is necessary to ensure that each item in a record (row of the database $\Sigma$) is different from its original value. NeNDS can also be applied to data sets with multiple duplicate entries. In cases where duplicates exist in a neighborhood, the permutation process ensures that each duplicate entry is substituted by a different non-identical item. The details of the substitution process in the presence of duplicates are explained later in this chapter.

Algorithm 5.2.1 shows the working of *NeNDS*. $\Sigma_{in}$ is the input database with $m$ attributes(fields) an $n$ records. The number of neighbors in each neighborhood, $c$, is the input parameter for the algorithm. Each individual dataset of the original and transformed database is denoted by ${\Sigma^i}_{in}$, ${\Sigma^i}_{out}$, respectively, where $i \in [1, m]$. Each dataset is divided into $NH_{size}$ neighborhoods, denoted by $NH_j$, $j \in [1, NH_{size}]$. The recursive **CreateTree** algorithm is then invoked to build a $c - ary$ tree for each $NH_j$. The procedure **Ancestors**(Tree, NH) returns all the ancestors of a specified node, and the procedure **Identical**(Parent, NH) returns all the entries in $NH$ that are identical to the parent of the specified node. *ChildrenTree* holds the set of valid children of the parent node in *Tree*. The populated tree is then assigned to the variable $Tree_j$ in Algorithm 5.2.1. All paths in

$Tree_j$ that have a length equal to the size of the neighborhood are candidates for substitution. The maximum edge distance $C_{ME}$ is determined for each candidate path. Procedure **min**(CandidateSet) identifies the path with the smallest $C_{ME}$ as the optimum substitution pattern. This path is then assigned to $NH'_j$. The datasets $({\Sigma^1}_{out}, {\Sigma^2}_{out}, \ldots, {\Sigma^m}_{out})$ form the transformed database $\Sigma_{out}$. Each dataset ${\Sigma^1}_{out}$ is reconstructed from the transformed neighborhoods as shown in Equation 5.

$${\Sigma^i}_{out} = (NH'_1,\ NH'_2,\ \ldots,\ NH'_{NHsize}) \tag{5}$$

### 5.2.1 Algorithm

**NeNDS(c)**

1. *For each* $i \in [1, m]$ *do*
   (a) $NHsize = \lfloor \frac{N}{c+1} \rfloor$
   (b) ${\Sigma^i}_{in} = (NH_1, NH_2, \ldots NH_{NHsize})$
   (c) *For each* $NH_j \in {\Sigma^i}_{in}$ *do*
      i. $Tree_j$ = **CreateTree**($NH_j$,0, NHsize)
      ii. $d_j = depth(Tree_j)$
      iii. *For each* $path_k$ in $Tree_j$ of length $d_j - 1$
         - CandidateSet = CandidateSet + $(path_k)$
      iv. $NH'_j$ = **min**(CandidateSet)
2. ${\Sigma^i}_{out} = (NH'_1, NH'_2, \ldots, NH'_{NHsize})$

**CreateTree(NH, Tree, Size)**

1. *If* $Tree = 0$ *then* Tree = NH[0]
2. *If* $NH = 0$ *then Return* Tree
3. ChildrenTree = NH - **Ancestors**(Tree) - **Identical**(Parent, NH)
4. Child(Tree) = **Sort**(ChildrenTree)

5. Tree = Child(Tree)

Figure 4 shows the tree-based permutation process for the neighborhood [35 37 38 40 42]. The first item in the set is made the root of the tree. The remaining elements [37 38 40 42] become the children of the root node and are ordered from left to right based on their nearness to the parent node. The distance between the parent and child is given along the edge connecting them. For instance, the edge distance between the root (35) and its left most child (37) is 2, which is assigned to the edge connecting the two nodes $E_{35-37}$ and is called the cost of the edge $C_{E_{35-37}}$. Each child of the root becomes the root of a sub tree with all the data items that are not in the path from the root of the tree to the root of the sub tree becoming the children of this root. The nodes are expanded using a depth first approach from left to right, which means that the leftmost child of the root is expanded completely before the next child is expanded. In the figure, the first child (37) of the root (35) becomes the parent node for the nodes [38 40 42], which are not in its path from the root. The next node that is expanded is (38) because it is the leftmost child of the parent (37). The leftmost child is expanded each time until the leaf node is reached. The node (42) is the leaf node for the leftmost path in the figure. The root node is appended to the leaf and the distance between the two nodes is assigned as its edge distance. The edge with the largest edge cost, called $C_{ME}$ is marked with a green box. This is the maximum cost of the specified path. The path ($35 \rightarrow 37 \rightarrow 38 \rightarrow 40 \rightarrow 42 \rightarrow 35$) with $C_{ME} = 7$ becomes a candidate for the permutation set. The next node that is expanded is the second child of the node (38), which is (42). The maximum edge cost for this path is $C_{ME} = 5$, which is less than the cost of the previous path and hence replaces the first path as the candidate permutation set. The new candidate set is now ($35 \rightarrow 37 \rightarrow 38 \rightarrow 42 \rightarrow 40 \rightarrow 35$) with $C_{ME} = 5$. The algorithm backtracks to the next unexpanded node, which is the second child of node (37), which is (40). The maximum edge cost for the leftmost path of this sub tree is $C_{ME} = 7$. Since the cost of the maximum edge for this path is greater than the cost of the candidate set, this path is ignored. The next path in the sub tree has an edge cost $C_{ME} = 3$, which is smaller than the maximum cost of the candidate set. The path ($35 \rightarrow 37 \rightarrow 40 \rightarrow 42 \rightarrow 38 \rightarrow 35$) replaces the candidate set and becomes the new candidate

with $C_{ME} = 3$. The next node to be expanded is the third child of (37), which is (42). The edge cost for this edge $C_{E_{37-42}} = 5$, which is larger than $C_{ME}$. Any path in this sub tree would have a maximum edge cost greater than or equal to 5. Hence the node is aborted and marked in yellow. The rest of the children of the root node have edge costs greater than the maximum edge cost of the candidate set $C_{ME} = 3$. As a result, none of these nodes are expanded. The final candidate set is $(35 \rightarrow 37 \rightarrow 40 \rightarrow 42 \rightarrow 38 \rightarrow 35)$, which is represented by a green arrow in the figure. The permutation set for the neighborhood [35 37 38 40 42] is obtained from the candidate $35 \rightarrow 37 \rightarrow 40 \rightarrow 42 \rightarrow 38 \rightarrow 35$ by replacing each item in the neighborhood by the item on the right of it in the candidate set. The permutation set for this neighborhood is [37 40 35 42 38].

By considering only those nodes that are not yet in the path from the root of the tree to the root of the sub tree, the size of the tree is significantly reduced. This method also avoids swapping of data items. The maximum edge cost helps remove all those paths that would result in a less optimal solution, which reduces the complexity of the search process. Only those nodes that can lead to a minimum cost path are expanded. The depth first approach used here is derived from the Depth First Search tree traversal algorithm. Since the tree is finite, the DFS search will complete and will yield a solution.

The process of NeNDS based obfuscation is described with the help of an example database shown in Table 1. The database $\Sigma = [\textbf{Age Salary Location}]$ has three fields and 5 records. The database is first divided into 3 individual datasets $\Sigma_1 = \textbf{Age}$, $\Sigma_2 = \textbf{Salary}$, and $\Sigma_3 = \textbf{Location}$. Since the third dataset is a categorical set, it is not obfuscated. Only the **Age** and **Salary** fields are obfuscated. The minimum neighborhood size for NeNDS is 3, so the maximum number of neighborhoods into which the data sets can be divided is $NH = \lfloor \frac{5}{3} \rfloor = 1$. Both the data sets are treated as single neighborhood sets. The permutation process for the **Age** data set is shown in Figure 4 and the permutation set is represented in the first column of Table 2. The permutation tree for the **Salary** data set is shown in Figure 5. The items in the tree represent the first two digits of the salary, i.e. $88 = 88,000$. The candidate set for permutation for the **Salary** data set is $(85 \rightarrow 86 \rightarrow 93 \rightarrow 94 \rightarrow 88 \rightarrow 85)$. The permuted set is represented in the second column

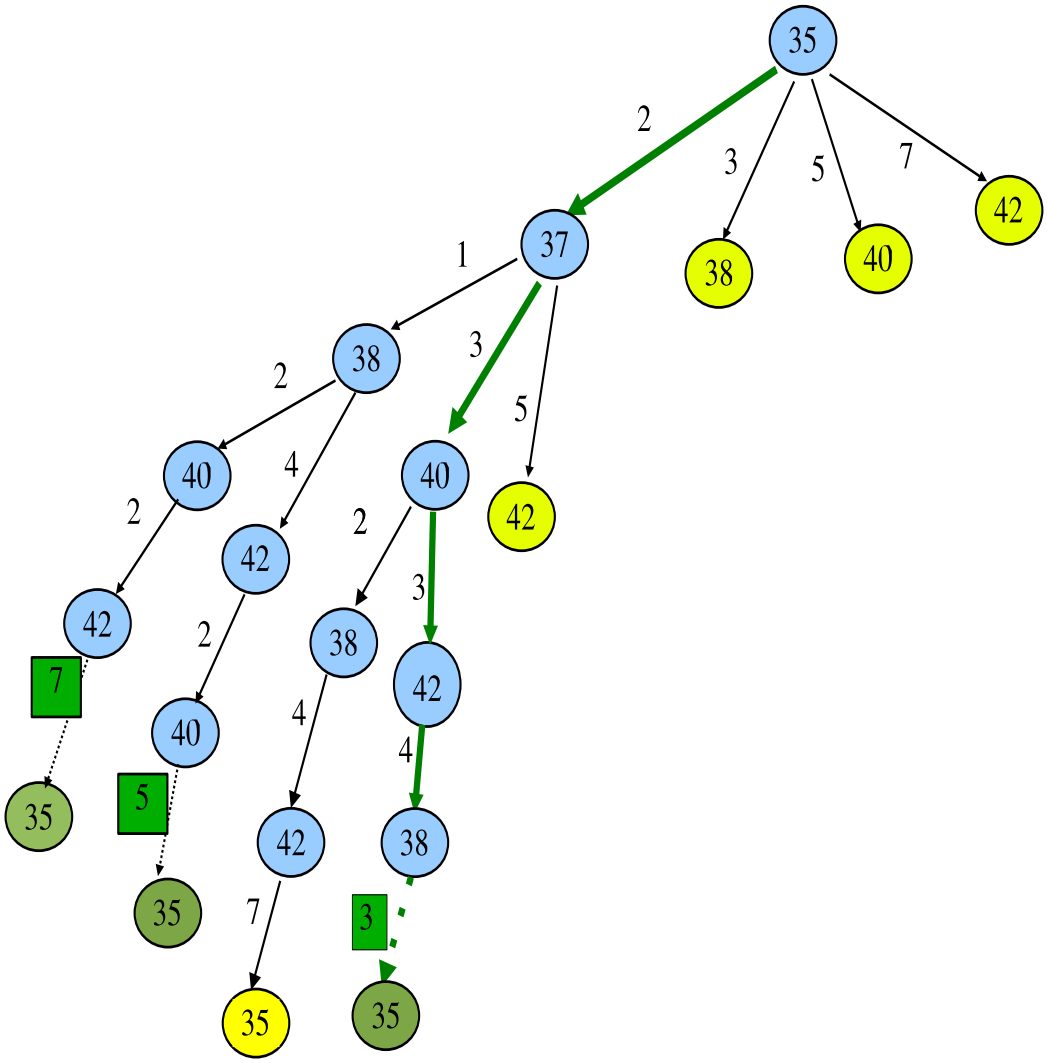



**Figure 4:** Permutation tree for the Age data set.

of Table 3.

Table 3 represents the completely transformed database. It can be observed from the two tables that the data transformation obscures the individual information in each record. Each record in the obfuscated database $\Sigma'$ is different from all the records in the original database $\Sigma$.

**Table 1:** Original database.

| Sl. No | Age | Salary | Location |
|---|---|---|---|
| 1 | 35 | 86,000 | LA |
| 2 | 37 | 88,000 | NY |
| 3 | 38 | 93,000 | SJC |
| 4 | 40 | 85,000 | SFO |
| 5 | 42 | 94,000 | LA |

*NeNDS* can be performed on any data set in which the elements are related by some notion of distance, and can be expressed as a metric space. The algorithm is run for each field in the database that forms a metric space. A brute-force analysis of the DFS based algorithm for finding the substitution pattern indicates that the algorithm has an exponential order of complexity. However, the heuristic nature of the branch and bound implemented reduces the exponential order of complexity to a much smaller value, which is indicated by the successful completion of *NeNDS* even for large data sets. The time complexity for NeNDS is discussed in Section 5.2.3

The data-substitution process for neighborhoods that contain multiple identical elements is the same as described above, with the additional restriction on identical element substitution - a parent node cannot have an identical child. The existence of such a path with no identical substitutions is ensured by including a sufficient number of non-identical elements in each neighborhood.

Clustering techniques can be used for creating neighborhoods of the datasets. The neighborhoods created are likely to be of different sizes, depending on the number of identical elements in each neighborhood. An additional condition is imposed on the size of a neighborhood with multiple identical entries - the neighborhood should contain at least one

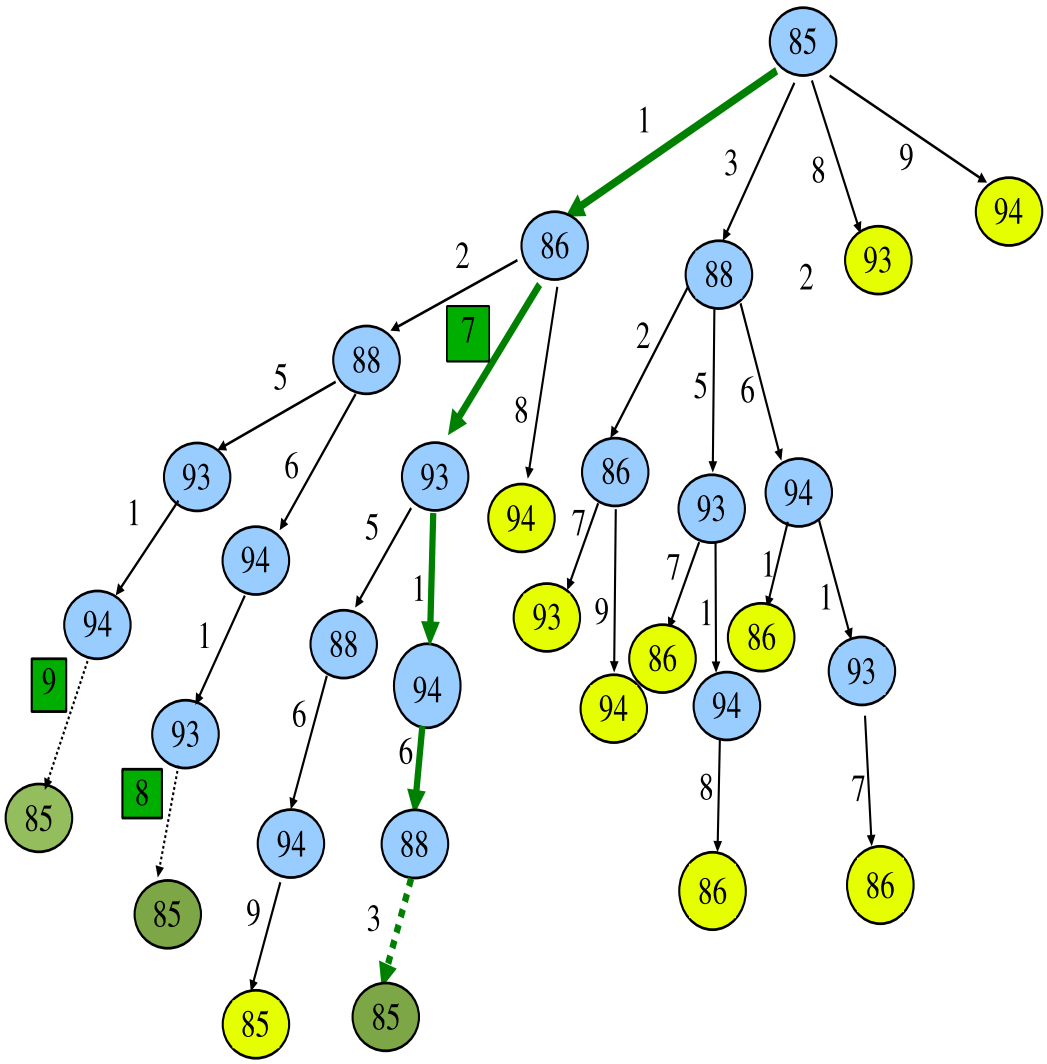



**Figure 5:** Permutation tree for the Salary data set.

**Table 2:** Age transformed database.

| Sl. No | Age | Salary | Location |
|---|---|---|---|
| 1 | 37 | 86,000 | LA |
| 2 | 40 | 88,000 | NY |
| 3 | 35 | 93,000 | SJC |
| 4 | 42 | 85,000 | SFO |
| 5 | 38 | 94,000 | LA |

**Table 3:** Age and Salary transformed database.

| Sl. No | Age | Salary | Location |
|---|---|---|---|
| 1 | 37 | 93,000 | LA |
| 2 | 40 | 85,000 | NY |
| 3 | 35 | 94,000 | SJC |
| 4 | 42 | 86,000 | SFO |
| 5 | 38 | 88,000 | LA |

distinct data element for each identical entry in the neighborhood. This requirement is enforced to ensure that each of the identical elements is substituted with a non-identical value. This rule can lead to the selection of very large neighborhoods in the event of several occurrences of identical elements. An alternative for obfuscating data sets with large number of duplicate data items is discussed in Section 5.4.1.

The algorithm treats is capable of handling multiple duplicate items in a neighborhood. When multiple duplicate items exist within a neighborhood, each duplicate is considered as a separate entry. The children of a node include only non-identical items. By ensuring no-swaps between duplicates, an optimal permutation set can be obtained for data sets with multiple duplicate entries too. Table 4 consists of a database with 9 records and 2 fields [**Age, Salary**].

Three records in the database have the same value for **Age**. The minimum size of a neighborhood ($NH_{size}$) with duplicate entries is three times the total number of duplicate data items. If $NH_{size}$ is between two and three times the total number of duplicate

**Table 4:** Original database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 35 | 75,000 |
| 2 | 35 | 80,000 |
| 3 | 35 | 78,000 |
| 4 | 38 | 81,000 |
| 5 | 39 | 120,000 |
| 6 | 42 | 110,000 |
| 7 | 44 | 105,000 |
| 8 | 48 | 130,000 |
| 9 | 50 | 125,000 |

entries, a permutation set is achievable, but one or more pairs of items would undergo swapping. Since the **Age** data set has three duplicates, the minimum size of the neighborhood needs to be 9, which includes the entire data set. The tree in Figure 6 shows the permutation tree for obfuscating the **Age** data set. The permutation set for this data set is [38 42 48 39 35 44 35 50 35]. The **Salary** data set is also permuted as a single neighborhood and the permutation set obtained is shown in the second column of Table 5. The algorithm described in this section is capable of handling a small number of multiple duplicate entries. Data sets that consist of significantly large number of duplicates are discussed in Section 5.4.

### 5.2.2 Limitation of NeNDS

The minimum size of a neighborhood for NeNDS is dependent on the existence and number of duplicate items. In the presence of $d$ duplicate items, the neighborhood size is increased to include at least $3d$ distinct values. This dependency limits the maximum number of duplicate items in a dataset to one-third of the entire dataset. This makes NeNDS unsuitable for binary valued fields or categorical fields.

The nearest neighbor permutation for NeNDS uses the distance between data items to determine the nearest distance neighbors. NeNDS-based data obfuscation can be only be applied to datasets that form a metric space. Hence, attributes such as names and

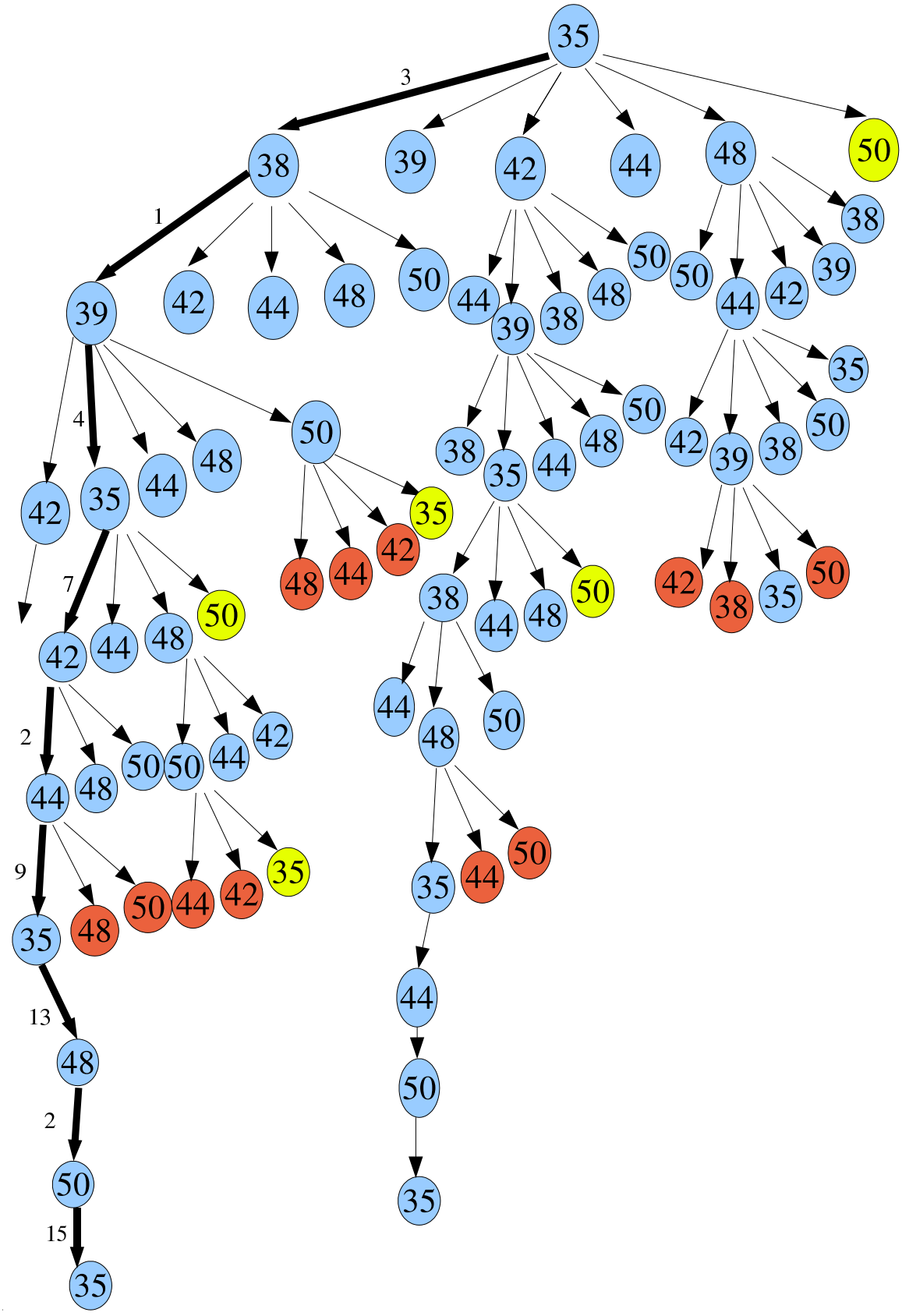



**Figure 6:** Illustration of permutation tree with duplicate entries.

**Table 5:** Obfuscated database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 38 | 80,000 |
| 2 | 42 | 78,000 |
| 3 | 48 | 81,000 |
| 4 | 39 | 75,000 |
| 5 | 35 | 125,000 |
| 6 | 44 | 105,000 |
| 7 | 35 | 120,000 |
| 8 | 50 | 110,000 |
| 9 | 35 | 130,000 |

locations cannot be obfuscated using NeNDS. This problem can be averted by performing data generalization or suppression on the non-numeric and categorical attributes and a data substitution on the rest of the database. This ensures a completely robust framework for data mining applications by preserving all the information content for statistical analysis purposes, and providing a secure and privacy-preserving framework for drawing inferences on the data.

### 5.2.3 Time Complexity of NeNDS

NeNDS uses a Depth First Search (DFS) tree traversal approach with backtracking to determine the optimal permutation set for each neighborhood in the data set that is to be obfuscated. The permutation tree shown in Figure 5 is used as an example to analyze the complexity of NeNDS. Each path in the tree is a potential candidate for substitution as it has a depth equal to the neighborhood size. The algorithm proceeds by traversing each path one at a time with backtracking. The first path in this tree ($85 \rightarrow 86 \rightarrow 88 \rightarrow 93 \rightarrow 94 \rightarrow 85$) has a max edge cost $C_{ME} = 9$. The second path that is traversed, ($85 \rightarrow 86 \rightarrow 88 \rightarrow 94 \rightarrow 93 \rightarrow 85$), has a $C_{ME} = 7$. The third path, ($85 \rightarrow 86 \rightarrow 93 \rightarrow 88 \rightarrow 94 \rightarrow 85$) has a cost $C_{ME} = 9$ and is eliminated, but the fourth path ($85 \rightarrow 86 \rightarrow 93 \rightarrow 94 \rightarrow 88 \rightarrow 85$) with $C_{ME} = 7$ is considered as a potential candidate. The sub tree for the next child of the root is truncated at the second or third levels because the edge costs exceed the current $C_{ME}$.

It can be seen that the edge costs between the root node and all the remaining children of the root are greater than $C_{ME}$, so the search is terminated. The optimal permutation set for this neighborhood is [93 85 94 86 88]. In this case, the permutation set was obtained by expanding less than half the nodes in the set. In the worst case, every path of the pruned tree needs to be traversed before the ideal permutation set is obtained. The time required for exploring each node of the tree is $(n-1)!$. Although the worst case time complexity for the algorithm has a complexity $O((n-1)!)$, a solution is obtained by exploring less than half the tree in most cases. Since each of the paths in the tree is a candidate set for permutation, the algorithm is guaranteed to produce a result always.

## *5.3 GT-NeNDS: A Hybrid Data Substitution Approach*

NeNDS obfuscates data by permuting sets of similar items. The example provided in Tables 1, 2, and 3 shows how the data is obfuscated. The original values of the data are modified, however, these values are still close enough to the original values to be considered a breach of privacy. While the proximity to the original data is suitable for some applications, it is insufficient to provide privacy in highly sensitive databases. Increasing the distance between the original and obfuscated value ensures better privacy but results in loss of information content. NeNDS obfuscates data without "modifying" the values of the data set. Items in each data set are replaced by other similar items so that the information stored in each record of the database is not distorted significantly. The advantage of obfuscating the data in this way is that the clusters in the original database are not altered even after obfuscation. An enhancement to the basic NeNDS algorithm, which protects against the vulnerability due to "proximity breaches" is proposed in this section.

In this section, we propose a hybrid version of *NeNDS*. In this approach, termed as *GT-NeNDS*, the data sets are transformed using NeNDS and then geometrically transformed. Here, the geometric transformations provide an additional privacy wrapper around NeNDS. The transformation functions like rotation and translation are isometric in nature, thereby preserving cluster information of the data sets and retaining the nearest neighbor information for the substitution step. In this way, the data can be transformed to a form suitable

for use by a third party analyst. The two step transformation results in transformed data that preserve clustering information, but bear no resemblance to the original database. As a result, *GT-NeNDS* is also robust to the notion of privacy breaches as proposed by [20], making it a suitable candidate for privacy preserving data mining.

The procedure for GT-NeNDS is explained with the example database shown in Table 6. The database $\Sigma = [\textbf{Age Salary}]$ has two fields and 9 records. Each individual data set is first obfuscated using NeNDS. Since the **Age** data set has 3 duplicate entries, the minimum size of the neighborhood for the data set is 1. The salary field is divided into 2 neighborhoods $NH_1 = [75,000\,78,000\,80,000\,81,000]$ and $NH_2 = [105,000\,110,000\,120,000\,125,000\,130,000]$. The **Age** transformed database is shown in Table 7. The permutation sets for each of the neighborhoods in the **Salary** data set are $NH'_1 = [80,000\,75,000\,81,000\,78,000]$ and $NH'_2 = [120,000\,105,000\,130,000\,110,000\,125,000]$. The complete NeNDS obfuscated database is shown in Table 8.

The NeNDS-obfuscated database can be geometrically transformed by using either rotation or scaling. The obfuscated databases using both scaling and rotational transformations are represented here. For scaling, the first data set is scaled by 1.4 and the second data set is scaled by 0.8. For rotational data transformation, the database is rotated by an angle of $30^o$. Rotation is performed by first normalizing the data sets. Normalization is done in order to reduce the data sets to the same scale before rotation. Each item $D_{\Sigma_i}$ in a data set $\Sigma_i$ is normalized using Equation 6, where Mean($\Sigma_i$) is the mean of the data set and Var($\Sigma_i$) is the variance of the items in the data set.

$$D_{\Sigma_i normalized} = \frac{|D_{\Sigma_i} - Mean(\Sigma_i)|}{Var(\Sigma_i)} \tag{6}$$

The scaled and rotated databases are shown in Tables 8 and 9. The databases bear no resemblance to the original database and therefore protect the privacy of the records in the database.

This type of obfuscation provides robust privacy protection to the database. The linearity of the geometric transformations ensures that the inter-relationships of the data items

**Table 6:** Original database for GT-NeNDS.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 35 | 75,000 |
| 2 | 35 | 80,000 |
| 3 | 35 | 78,000 |
| 4 | 38 | 81,000 |
| 5 | 39 | 120,000 |
| 6 | 42 | 110,000 |
| 7 | 44 | 105,000 |
| 8 | 48 | 130,000 |
| 9 | 50 | 125,000 |

**Table 7:** GT-NeNDS: NeNDS Obfuscated database.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 38 | 80,000 |
| 2 | 42 | 81,000 |
| 3 | 48 | 75,000 |
| 4 | 39 | 78,000 |
| 5 | 35 | 130,000 |
| 6 | 44 | 105,000 |
| 7 | 35 | 120,000 |
| 8 | 50 | 125,000 |
| 9 | 35 | 110,000 |

in the database are preserved. The data items are scaled and rotated while preserving the distances between the individual items. The clusters of similar items are preserved even after transformation. NeNDS combines data items that are similar and permutes the set of similar items. The information contained in each record of the database is distorted only by a small amount. The information provided by each record remains approximately the same. Records that are similar to each other in the original database preserve the similarity property even after being transformed by NeNDS or GT-NeNDS. This is because each item in the record is replaced by another item that is similar to it. The cluster preserving

**Table 8:** GT-NeNDS: Scaled NeNDS transformed database.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 53 | 64,000 |
| 2 | 59 | 64,800 |
| 3 | 67 | 60,000 |
| 4 | 55 | 62,400 |
| 5 | 49 | 104,000 |
| 6 | 62 | 84,000 |
| 7 | 49 | 96,000 |
| 8 | 70 | 100,000 |
| 9 | 49 | 88,000 |

**Table 9:** GT-NeNDS: Rotated NeNDS transformed database.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 41 | 3,980,000 |
| 2 | 41 | 1,965,600 |
| 3 | 42 | 10,930,000 |
| 4 | 41 | 2,492,000 |
| 5 | 42 | 8,448,000 |
| 6 | 41 | 4,973,400 |
| 7 | 42 | 4,937,400 |
| 8 | 42 | 13,908,000 |
| 9 | 42 | 8,447,700 |

property makes NeNDS and GT-NeNDS favorable for data mining applications.

## *5.4 NeNDS: Special Cases*

NeNDS obfuscates data by creating neighborhoods of similar items, generating a permutation set for each neighborhood, and replacing the original neighborhoods with their corresponding permutation sets. The generation of a permutation set has been explained in the previous section. While the algorithm provides a general solution for nearest neighbor data substitution, there are a few cases where the data need to be treated differently. Two special cases are discussed in this section.

### 5.4.1 Duplicate data items

The Algorithm for NeNDS generates a permutation pattern for obfuscating data sets with unique data items as well as data sets with a small number of duplicate entries. However, if the number of duplicate entries is significantly large, the data are treated differently. Consider the example shown in Table 10. The **Age** field in the database has more than half of the entries with the same value for age. In such cases, the identical entries are left unobfuscated. Only the unique data items are obfuscated. The obfuscated **Age, Salary** data sets are shown in Table 11. It can be observed that the salaries corresponding to the duplicate age records have changed although the ages are still the same. If necessary, the duplicates are obfuscated using data randomization. Such situations are discussed in Section 5.4.2. The large number of duplicates makes the records indistinguishable and the additional distortions of the rest of the fields protect the identities of the records.

**Table 10:** Original database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 35 | 75,000 |
| 2 | 37 | 80,000 |
| 3 | 38 | 81,000 |
| 4 | 39 | 85,000 |
| 5 | 39 | 78,000 |
| 6 | 39 | 110,000 |
| 7 | 39 | 105,000 |
| 8 | 40 | 120,000 |
| 9 | 42 | 115,000 |

The database can be further protected by applying a geometric transformation to the NeNDS transformed database. The geometric transformation masks the values of the identical items completely. Table 12 shows a scaled database with a scaling factor of 1.3 for each data set.

In general, if the number of duplicates in the data set exceeds a pre-specified value $k$, the duplicates can be left unobfuscated. The value $k$ is an input parameter to be entered

**Table 11:** Obfuscated database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 37 | 80,000 |
| 2 | 40 | 78,000 |
| 3 | 35 | 85,000 |
| 4 | 39 | 81,000 |
| 5 | 39 | 75,000 |
| 6 | 39 | 125,000 |
| 7 | 39 | 115,000 |
| 8 | 42 | 105,000 |
| 9 | 38 | 110,000 |

**Table 12:** Scaled-NeNDS transformed database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 48 | 104,000 |
| 2 | 52 | 101,400 |
| 3 | 46 | 110,500 |
| 4 | 51 | 105,300 |
| 5 | 51 | 97,500 |
| 6 | 51 | 162,500 |
| 7 | 51 | 149,500 |
| 8 | 55 | 136,500 |
| 9 | 49 | 143,000 |

by the database administrator. The default value of $k$ is one-third of the size of the data set. Hybrid-NeNDS protects the actual value of the duplicates by applying a geometric transformation to the actual values.

### 5.4.2 Identical transformations

NeNDS obfuscates each data set of the database individually by selecting neighborhoods of similar items and permuting the data in each neighborhood. In this type of obfuscation there is a likelihood of one or more data items of a specific record to be transformed in the same way. This would result in the record remaining identical except for a change of position. Table 13 shows a database $\Sigma$ = [**Age Salary**] with 5 records. The NeNDS obfuscated database is shown in Table 14, the first record in the original database is identical to the third record in the obfuscated database. A post processing step for NeNDS involves scanning the entire database to ensure that the records in the obfuscated database do not map back to any record in the original database. In case of a match, one of the data sets is replaced by a different permutation set as shown in Table 15. The data items have been permuted such that each of the records in the output database is different from all records in the original database.

**Table 13:** Special case: Original database.

| Sl. No | Age | Salary | Location |
|---|---|---|---|
| 1 | 35 | 76,000 | LA |
| 2 | 37 | 79,000 | NY |
| 3 | 38 | 84,000 | SJC |
| 4 | 40 | 78,000 | SFO |
| 5 | 42 | 82,000 | LA |

The probability that such situations should occur is discussed here. The database shown in Table 13 has only two data sets. Each item in the first data set can be moved to 4 other positions. Similarly, each item in the second data set can be moved to 4 other positions. The probability that both items in any row $r_i$ in the original database are moved to the row $r_j,\ i \neq j$ in the obfuscated database is $P(true) = 1/4 * 1/4 * 5 = 0.31$. In general, for

**Table 14:** Special case: Obfuscated database.

| Sl. No | Age | Salary | Location |
|---|---|---|---|
| 1 | 37 | 84,000 | LA |
| 2 | **40** | **78,000** | **NY** |
| 3 | 35 | 82,000 | SJC |
| 4 | 42 | 84,000 | SFO |
| 5 | 38 | 79,000 | LA |

**Table 15:** Special case: Re-obfuscated database.

| Sl. No | Age | Salary | Location |
|---|---|---|---|
| 1 | 37 | 78,000 | LA |
| 2 | 40 | 76,000 | NY |
| 3 | 35 | 82,000 | SJC |
| 4 | 42 | 84,000 | SFO |
| 5 | 38 | 79,000 | LA |

a database $\Sigma = [\Sigma_1, \Sigma_2, \cdots, \Sigma_m]$ with $m$ fields (data sets) and $N$ records, the probability that the obfuscation results in moving all items of a row in the same way is given as $P = (\frac{1}{N-1})^m N$. The probability of identical transformations decreases significantly for databases with more fields. Practical databases have $10 - 100$ fields on an average, which reduces the probability of identical transformations considerably.

A special case involving identical transformations is when a record is not displaced from its original position even after data obfuscation using NeNDS or GT-NeNDS. For data sets with no duplicate items, NeNDS ensures that each data item is displaced from its original position. This is also holds true when the number of duplicate items in a data set is small in comparison with the size of the data set. However, if there are a large number of duplicates, the NeNDS based DO obfuscates the unique items and leaves the duplicate items unobfuscated as discussed in Section 5.4.1. In this case, the duplicate data item is not displaced from its original position in the database even after obfuscation. The likelihood of an entire record remaining unchanged is demonstrated with the help of the database shown in Table 16. The database $\Sigma =$[**Age Salary**] consists of two fields and 9

records. The minimum neighborhood size for this neighborhood is 4. The **Age** data set has 5 duplicate items, which is greater than 1/3 of the data set. The data set is divided into two neighborhoods. The first neighborhood consists of the 4 duplicate items and the second data set includes the unique data items. The **Salary** data set has 5 duplicate items. The number of duplicate items in this data set is also grater than 1/3 of the data set. This data set is therefore divided into two neighborhoods: the first neighborhood includes all the unique items and the second neighborhood consists of the duplicate data items. The result of the NeNDS-based DO is shown in Table 17.

**Table 16:** Special case: Original database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 35 | 86,000 |
| 2 | 35 | 89,000 |
| 3 | 35 | 94,000 |
| 4 | 35 | 88,000 |
| 5 | 35 | 105,000 |
| 6 | 37 | 92,000 |
| 7 | 38 | 105,000 |
| 8 | 40 | 105,000 |
| 9 | 42 | 105,000 |

**Table 17:** Special case: Obfuscated database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 35 | 88,000 |
| 2 | 35 | 92,000 |
| 3 | 35 | 86,000 |
| 4 | 35 | 89,000 |
| 5 | **35** | **105,000** |
| 6 | 40 | 94,000 |
| 7 | 37 | 105,000 |
| 8 | 42 | 105,000 |
| 9 | 38 | 105,000 |

It can be seen that the fifth record in the original database and the obfuscated database is the same, leaving record 5 unprotected against privacy attacks. This situation is handled by re-obfuscating some of the neighborhoods in each data set in the following way.

In the first data set, the number of data items in the duplicate-items-neighborhood is 5. In this case, the duplicate data item corresponding to row 5 is combined with the second neighborhood. The sizes of the two neighborhoods for the **Age** data set are now $NH1_{size} = 4$ and $NH2_{size} = 5$, where $NH_1$ is the duplicate items set and $NH_2$ is the neighborhood with unique data items. The corresponding obfuscated data set for the **Age** data set is shown in the first column of Table 18.

In the second data set, the number of duplicate items is 4. If the data item corresponding to row 5 is moved to the neighborhoods with unique data items, $NH_1$, then the neighborhood size of the duplicate-items-neighborhood is reduced to 3, which is lesser than the minimum size specified for a neighborhood $NH_{size} = 4$. In this case, all the duplicate items are randomized by adding a noise factor generated from a Normal (Gaussian zero-mean) distribution $[0, \delta]$, where $\delta$ is the deviation of the data items in the neighborhood containing unique data items. The deviation $\delta$ for this example is $\delta = 3.89$. The noise added neighborhood is shown in the second column of Table 18.

**Table 18:** Special case: Re-obfuscated database with duplicates.

| Sl. No | Age | Salary |
|---|---|---|
| 1 | 35 | 88,000 |
| 2 | 35 | 92,000 |
| 3 | 35 | 86,000 |
| 4 | 35 | 89,000 |
| 5 | 37 | 106,000 |
| 6 | 40 | 94,000 |
| 7 | 35 | 101,200 |
| 8 | 42 | 111,600 |
| 9 | 38 | 102,000 |

As a generalization, let $p$ be the number of data items in the duplicate-items-neighborhood

that have to be re-obfuscated. Let $NH_{dup_{size}}$ be the number of duplicate items in a duplicate-items-neighborhood. If $NH_{dup_{size}} - p > NH_{size}$, then the $p$ data items are moved to the next closest neighborhood. The new neighborhood is permuted with the new data items that are included. If the difference $NH_{dup_{size}} - p < NH_{size}$, then all the $NH_{dup_{size}}$ data items are obfuscated by adding a random noise factor generated from a normal distribution $[0, \delta]$. The value of $\delta$ is computed according to Equation 7, where $\mu_{NH_{next}}$ is the mean of the data items in the next closes neighborhood and $X$ is the value of the duplicate data items.

$$\delta = 1/2(\mu_{NH_{next}} - X) \tag{7}$$

The $p$ offsets generated from the normal distribution are added to the duplicate items. All the identical transformations resulting from duplicate data items are treated in this way.

The situations discussed in this section show that NeNDS is capable of handling situations that might lead to a privacy breach due to identical transformations.

# CHAPTER VI

# EVALUATING PRIVACY AND USABILITY OF DO TECHNIQUES

The working mechanisms of different data obfuscation (DO) techniques including NeNDS-based DO are described in Chapter 5.2. This chapter compares the strengths and weaknesses of the different DO techniques by evaluating their ability to provide robust privacy protection and also preserve the usability of the databases after obfuscation. The privacy protection capability is evaluated based on the privacy properties discussed in Chapter 4.3. The Usability comparison is done experimentally by evaluating the ability of the DO techniques to obfuscate the databases without distorting the inherent clusters among the data items.

## *6.1 Privacy Analysis*

The privacy protection provided by DO techniques are evaluated based on two important properties, difficulty of reverse-engineering the obfuscated database and the nonresemblance of the obfuscated database to the original database. This section analyzes some of the DO techniques based on their privacy properties.

### 6.1.1 Analysis of Random Data Perturbation

Random data perturbation techniques use random noise to obfuscate the data. The randomness makes the obfuscated database irreversible. The nonresemblance property is dependent on the parameters used to generate the Gaussian or uniform noise distribution. If the noise distribution is generated with a small deviation from the mean, the resulting obfuscated data is similar to the original data, making the DO vulnerable to approximate privacy breach. Nonresemblance of the obfuscated database is achieved by adding a noise distribution generated with a large deviation. The maximum deviation of the noise distribution is

limited by the usability property, which is evaluated in Section 6.2.

### 6.1.2 Analysis of Geometric Transformations

Geometric transformations fall under the category of linear transformation functions. These functions are susceptible to *partial reversibility* and are the most vulnerable DO techniques. The linearity property of this data obfuscation technique preserves the clustered nature of the data, but also results in weak privacy protection. The assumption made here is that the attacker is aware that the DO process is a linear transformation. In this case, we prove that for a database with $d * n$ entries, where $d$ is the number of attributes and $n$ is the number of records, the knowledge of only $d + 1$ affinely independent records in the original matrix is sufficient to uniquely determine the linear transformation. Once the transformation matrix is obtained, all the original data entries for which the obfuscated values are available are compromised. Therefore, the Geometric transformations of [44] [45], being instances of linear transformation functions, are compromised with the knowledge of $d + 1$ affinely independent records in the original data.

In the proof of Theorem 6.1.2, matrix $A$ is the transformation matrix and $x, y$ represent the original and obfuscated database. Each $x_i, y_i : i \in 1 \dots k$, represent the values in one attribute of the original and obfuscated databases respectively. The proof shows that with the knowledge of only $k + 1$ records in the original database, the rest of the data can be retrieved. The affine independence condition ensures that the data space is completely covered. Because of this condition, if the records are affinely dependent, more than $k + 1$ of them are needed to determine the transformation function. As the number of records needed for complete reversal is a function of the number of attributes involved in the data-obfuscation process, a database in which only a few of the attributes are obfuscated would be easier to reverse than one in which more of the attributes are involved in the obfuscation.

**Theorem 1.** *Let* $f(x) = Ax + b : R^k \rightarrow R^k$ *where*

$$A = \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_k^T \end{pmatrix} \in R^{m*k} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix} \in R^k \tag{8}$$

*Suppose that*

$$\begin{array}{cccccc} x_1, & \ldots, & x_m & \in & R^k \\ y_1, & \ldots, & y_m & \in & R^k \\ Ax_1 & + & b & = & y_1 \\ \vdots & \vdots & \vdots & \vdots & \\ Ax_m & + & b & = & y_m \end{array} \tag{9}$$

*where $m = k + 1$, and the matrix given in (10) is invertible.*

$$\begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix} \tag{10}$$

*Let*

$$\overline{A} = \begin{pmatrix} \overline{a_1^T} \\ \overline{a_2^T} \\ \vdots \\ \overline{a_m^T} \end{pmatrix} \in R^{k*k} \quad \overline{b} = \begin{pmatrix} \overline{b_1} \\ \overline{b_2} \\ \vdots \\ \overline{b_m} \end{pmatrix} \in R^k \tag{11}$$

*If there exists any pair $\overline{A}$ and $\overline{b}$ that satisfies the set of equations given by (12), for $i = 1 \ldots k, j = 1 \ldots m$, then it can be proved that $\overline{A} = A, \overline{b} = b$ :*

$$\overline{a_i^T} x_{ij} + \overline{b_i} = y_{ij} \tag{12}$$

*Proof.* First show the existence of $\overline{A}, \overline{b}$, such that Equations (12) are satisfied.

$$\begin{array}{rccccc} & \overline{a_i^T} x_1 & + & \overline{b_i} & = & y_{i1} \\ \forall i & \vdots & & \vdots & \vdots & \vdots \\ & \overline{a_i^T} x_m & + & \overline{b_i} & = & y_{im} \end{array} \tag{13}$$

and $\overline{a_i}$ and $\overline{b_i}$ occur uniquely in Equation (13). For the big set of Equations (12) to have a solution, it is sufficient to show that (13) has a solution for each $i = 1 \ldots k$. $\forall i = 1 \ldots k \in$ (13), Equation (14) holds true.

$$\begin{pmatrix} x_1^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix} \quad \begin{pmatrix} \overline{a_i} \\ \overline{b_i} \end{pmatrix} \quad = \begin{pmatrix} y_i 1 \\ \vdots \\ y_i m \end{pmatrix} \tag{14}$$

This has a solution if (15) is invertible, which is an assumption.

$$\begin{pmatrix} x_1^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix} \tag{15}$$

Therefore, $\exists \overline{A}$ and $\overline{b}$ satisfying (12). Now we need to show that $\overline{A}x + \overline{b} = Ax + b \forall x \in R^k$. Since the matrix (15) is invertible, the matrix (15) form the basis for the vectors in the k*n original matrix. Hence, $\forall x \in R^k, \exists \alpha_1, \ldots \alpha_m \in R$, such that $x = \alpha_1 x_1 + \ldots + \alpha_m x_m$ and $\alpha_1 + \ldots + \alpha_m = 1$.

$$\begin{aligned} \overline{A}x + b \quad &= \quad \overline{A}(\sum_{j=1}^{m} \alpha_j x_j) + \overline{b}(\sum_{j=1}^{m} \alpha_j) \\ &= \quad \sum_{j=1}^{m} \alpha_j (\overline{A} x_j + \overline{b}) = \sum_{j=1}^{m} \alpha_j (y_j), \text{ since } \overline{A}, \overline{b} \text{ satisfy( 12)} \\ &= \quad \sum_{j=1}^{m} \alpha_j (A x_j + b) \\ &= \quad A(\sum_{j=1}^{m} \alpha_j x_j) + b(\sum_{j=1}^{m} \alpha_j) \\ &= \quad Ax + b \\ &\Rightarrow \quad \overline{A} = A \text{ and } \overline{b} = b \end{aligned} \tag{16}$$

*Q.E.D*

□

The proof of Theorem 6.1.2 indicates that linear transformation functions offer poor privacy. These techniques can transform the data items in such a way that they bear no resemblance the original data items. Geometric transformations offer weak privacy in terms of reversibility, but offer strong privacy against approximate privacy breaches.

### 6.1.3 Analysis of NeNDS and Data Swapping

Data swapping and *NeNDS* fall under the category of non-linear bijective transformations. In this type of transformation, *reversibility* is dependent on the minimum number of records $r$ that are sufficient for complete reverse engineering. In the case of data swapping, the minimum value for $r$ is half the number of elements in the data set. For each element in the data set that is known *a priori*, the corresponding element involved in the swap is revealed.

**NeNDS for distinct items**

In the case of *NeNDS*, complete reversal of the entire data set would require the knowledge of at least $r = c - 1$ distinct data elements for each neighborhood, where $c$ is the minimum size of a neighborhood. Even partial reversal of a single neighborhood would require the knowledge of $c - 1$ of its elements. The fraction $\frac{c_i - 1}{c_i}$ determines the ease of reversal of a specific neighborhood $i$ having exactly $c_i$ elements. The proof for this claim is provided below. The goal of the attacker is to retrieve the original value corresponding to one of the obfuscated items in a dataset with absolute certainty. We refer to this as a targeted value attack.

**Theorem 1.** *Let $[X, Y]$ be the original and obfuscated datasets of size $n$ respectively.*

$$X = x_1, x_2, \ldots, x_n \tag{17}$$

$$Y = y_1, y_2, \ldots, y_n \tag{18}$$

*Let $y_t | y_t \in Y$ be the obfuscated item whose original value $x_t$ the attacker wants to retrieve and let $x_t$ belong to the $p^{th}$ neighborhood. Assume that all $c$ items in the $p^{th}$ neighborhood are distinct values. Assume that the attacker has complete knowledge of the NeNDS algorithm,*

*including the value of neighborhood size c used to produce Y, but no additional knowledge except for a subset of the original data items. Then, the attacker needs to know at least* $c-1$ *original data items other than the targeted item to succeed in a targeted value attack.*

*Proof.* Let $[X_p, Y_p]$ be the original and obfuscated data items in the $p^{th}$ neighborhood.

$$X_p = x_{p1}, x_{p2}, \ldots, x_{pc} \tag{19}$$

$$Y_p = y_{p1}, y_{p2}, \ldots, y_{pc} \tag{20}$$

We evaluate what can be determined with the knowledge of at most $c-2$ original data items.

The only information known to the attacker:

$$X'_p = x_{p1}, x_{p2}, \ldots, U, \ldots, U, \ldots, x_{pc} \tag{21}$$

$$Y = y_1, y_2, \ldots, y_n \tag{22}$$

where $X'_p$ is a set of $c-2$ original data items, and each $U$ represents a missing value. The goal of the attacker is to identify two missing original values and determine which of these corresponds to the original value of $y_t$.

Case 1: There exist two items in the obfuscated dataset $y_k, y_l$ that fall within the interval $[min(Y_p), max(Y_p)]$. In this case, the attacker knows that $y_k, y_l$ are the missing items in the neighborhood $p$. These two items can be placed in the neighborhood in two ways, both of which produce the same obfuscated neighborhood $Y_p$:

$$X'_p = x_{p1}, x_{p2}, \ldots, y_k, \ldots, y_l, \ldots x_{pc} \tag{23}$$

$$X''_p = x_{p1}, x_{p2}, \ldots, y_l, \ldots, y_k, \ldots x_{pc} \tag{24}$$

Since there is no additional information that enables the attacker to accurately identify which of the two sequences $X'_p$, $X''_p$ is the original neighborhood, the attacker cannot determine with certainty whether $y_k$ or $y_l$ is equal to $x_t$.

Case 2: There are no items in the obfuscated data set that fall within the interval $[min(Y_p), max(Y_p)]$. In this case, the missing items are one of the three pairs: $min(Y_p) - 2, min(Y_p) - 1$, $max(Y_p) + 1, max(Y_p) + 2$ or $min(Y_p) - 1, max(Y_p) + 1$. For each pair, there are two permutations of the neighborhood that could be the original neighborhood. In this case, the original value corresponding to $y_t$ can be one of 6 values, and the attacker cannot determine with certainty which of these corresponds to $x_t$.

Case 3: One item in the obfuscated dataset lies in $[min(Y_p), max(Y_p)]$. Let this item be denoted as $y_{kl}$. In this case, the missing items can be one of two pairs: $min(Y_p) - 1, y_{kl}$ or $y_k l, max(Y_p) + 1$. Each pair can fill up the missing positions in two ways. In this case, there are 4 candidates corresponding to the original value for $y_t$ and again the attacker cannot know the value of $x_t$ with certainty.

This shows that even with the knowledge of $c - 2$ items in a neighborhood, the attacker cannot determine the original values of the remaining items with certainty.

□

**NeNDS with duplicates**

In the presence of duplicate entries, the minimum size of a neighborhood with $m$ duplicates is $c = 3m$. In this case, retrieving the original value of even a single obfuscated item requires *a priori* knowledge of at least $2m$ or $2c/3$ original items in the neighborhood containing the targeted original value. The minimum bound applies to cases where the unknown items are all duplicates. If the missing items are distinct, the minimum amount of information required is still $c - 1$ items in the original neighborhood $p$ that contains the targeted item. Even in the worst case for data with duplicate items, the attacker needs to know at least 2/3 of the items in a neighborhood to be able to retrieve even a specific targeted original value.

The poor privacy provided by linear transformation functions is proved in Section 6.1.2.

The cluster preserving property of the linear geometric transformations make them attractive for use in DO, but their vulnerability to reversal makes them unsuitable. The *NeNDS* transformation technique offers a stronger privacy preserving capability. In *GT-NeNDS*, The obfuscated data that results from Geometric transformations is obfuscated by *NeNDS*. Combining it with a stronger transformation function such as *NeNDS* strengthens the weak *reversibility* property of geometric transformations. The multi-tier obfuscation makes *GT-NeNDS* more difficult to reverse engineer than *NeNDS*. A comparison of the cluster retention capability is analyzed experimentally in Chapter 6.2, proving that *GT-NeNDS* is an optimum data obfuscation technique that provides robust data privacy as well as high data usability.

## *6.2 Experiment Results*

The reversibility analysis of *NeNDS* in Section 6.1.3 proves that this technique provides stronger privacy protection than other techniques like data swapping and geometric transformations. The *Usability* of data obfuscated using *NeNDS* is evaluated in this chapter. Usability refers to the usefulness of the obfuscated data for data mining applications. Data clustering techniques are commonly used in data mining applications to find patterns or similar trends in databases. The clustering results obtained by applying clustering techniques to obfuscated data are compared to the results obtained by applying the same techniques on the original data. The distortion introduced due to the obfuscated data is used to measure the usability of data obfuscation techniques.

### 6.2.1 Data Clustering Techniques

Data clustering techniques classify data into clusters or groups that are similar. Several data clustering techniques have been implemented for efficient clustering. These techniques are classified into two categories, hierarchical clustering and partitional clustering. Hierarchical clustering techniques, successive clusters are derived from parent clusters in an iterative manner. Partitional clustering techniques use a more direct approach and create

- Hierarchical clustering: Techniques that fall in this category build a hierarchy of

clusters. They begin by assuming each data item as a single cluster. Pairs of clusters that are closest to each other are then combined in the next step, to form clusters of size 2. This process is repeated until a single cluster is obtained, which includes all the data items. The clusters are represented as a tree with the single cluster as the root, and the single item clusters as the leaves. The rows of the tree represent the different clusters. Rows closer to the root provide coarser clusters and those further down the tree yield finer clusters with fewer elements. The *Elbow criterion* is commonly used to determine the row of the tree that yields the optimum number and size of clusters. The criterion states that the number of clusters should be chosen such that addition of another cluster does add significant value. In a graph plot of the percentage of variance versus the number of clusters, the point on the x-axis(number of clusters) corresponding to the elbow region of the graph indicates the optimum number of clusters. The distance measure used for combining data items is an important factor in hierarchical techniques. Commonly used distance metrics are:

  - Manhattan distance: This is also referred to as the 'city block' metric and is equal to the sum of absolute distances for each variable.
  - Euclidean distance: This distance measure is obtained by computing the square root of the sum of squares of the variables. The Euclidean distance is most commonly used as the distance measure for hierarchical clustering.

  Examples of hierarchical clustering techniques are the Single Linkage (SLINK), Complete Linkage (CLINK), and Group Average (GAVE). These techniques differ in the way the clusters are combined. SLINK combines clusters based on the shortest link between any two data items in each cluster. CLINK first determines the furthest points between pairs of clusters and combines the pair that has the smallest distance between the extreme items. GAVE forms clusters based on the average value of the items in each cluster rather than the minimum or maximum distances.

- Partitional clustering: Partitional clustering techniques differ from hierarchical techniques in that the number of clusters are decided *a priori*, and clusters are created

based on the specified number $M$ of clusters. Clusters are created by selecting $M$ representatives or cluster centers, following which all the data items are assigned to one of the cluster centers. The creation of cluster centers is dependent on the technique used. Examples of partitional clustering techniques are:

- K-means: The variable $K$ for the K-means algorithm is the number of clusters that are to be created. The algorithm randomly generates $k$ cluster centers and assigns each data item to the nearest cluster. The process of generating cluster centers and assigning the data items to each cluster is repeated until a bounding criterion is reached.
- Quality Threshold: In this algorithm, the user specifies the maximum distance between items instead of the number of clusters. The algorithm creates a cluster for each point by including items one at a time until the threshold distance is exceeded. The cluster containing the maximum number of items is preserved and all other clusters are dissolved. Clusters are created for each point that is not a part of the preserved luster and the process is repeated until there are no more items.
- Fuzzy c-means: This algorithm is a derivative of the K-means algorithm. In this technique, each item is associated with a probability co-efficient that indicated the degree of being in each cluster $k$.

The usability of the obfuscated data is evaluated by applying both types of clustering techniques on the data. The Group Average (GAVE) algorithm with Euclidean distance as the distance metric is used to evaluate the performance of hierarchical clustering. The performance of Partitional clustering techniques on the obfuscated data is studied by applying the K-means clustering algorithm.

### 6.2.2 Experimental Set-up

The evaluation of usability is carried out using both real and synthetic data. Real data is obtained from the UCI repository that provides sample data for data mining applications.

Two real databases are used in these experiments. The diabetes database containing 403 records and 19 that were collected to study the prevalence of obesity, diabetes, and other cardiovascular risk factors in central Virginia for African Americans. Of these 403 records, only 150 records were complete. The rest had missing entries in three or more fields. Hence, only the 150 complete records are used in the experimental analysis. Of the 19 fields, 3 fields are non-numerical and are not obfuscated. All the other fields are obfuscated. The Thyroid database containing $7,200$ records and 21 was collected to study symptoms of hypo- and hyper-thyroid conditions. Out of the 21 fields, 19 are binary fields and 6 fields are numeric-continuous fields. The six fields that are continuous and numeric are obfuscated.

Synthetic data is generated using an open-source synthetic data generator code that was developed as a part of IBM Almaden Research center's Quest Data generator project. The inherent clustering degree $C_{in}$ of the database to be generated can be specified as an input parameter, which enables the generation of databases with different clustering patterns. The other input specified to the data generator is the number of records required. The generator outputs a database with 9 fields and $N$ records, where $N$ is the number of records specified as the input.

1. Salary generated with equal likelihood probability in the range 20,000 to 150,000
2. Commission if Salary $\geq 75000$, Commission $= 0$ else as a number in the range $10000 - 75000$ with equal likelihood probability
3. Age in the range $20 - 80$
4. Education chosen from $0 - 4$
5. Car make of the car, chosen from $1 - 20$
6. Zipcode chosen from 9 available zipcodes
7. HouseValue a number in the range $0.5 * k * 100000 - 1.5 * k * 100000$ generated with equal likelihood probability, where $0 \leq k \leq 9$ and depends on the ZipCode
8. YearsOwned uniformly distributed from $1 - 30$

9. Loan uniformly distributed from $0 - 500,000$

Of these nine attributes, *Education, Car, Zipcode* are categorical attributes and are not considered for data obfuscation. The rest of the attributes are candidates for data obfuscation.

The original database is denoted as $DB_{orig}$. The set-up for each of the obfuscation techniques is as follows.

- NeNDS: Each dataset (field/attribute) of the database is divided into $NH$ neighborhoods. The minimum number of data items in each neighborhood $NH_{size}$ is an input parameter to the algorithm. The obfuscated data sets are recombined to form the obfuscated database, denoted as $DB_{NeNDS}$.

- Geometric Transformations: The transformation vector, rotation vector for rotation-based transformation, and a scaling vector for a scaling-based transformation are specified as inputs, denoted as $DB_{geom}$.

- Geometrically transformed NeNDS (GT-NeNDS): Each data set of the divided into $NH$ neighborhoods using NeNDS. The NeNDS obfuscated database is multiplied by a transformation vector to produce the GT-NeNDS obfuscated database, denoted as $DB_{GT-NeNDS}$.

The K-means clustering algorithm and the hierarchical Group Average algorithm are applied to each of the databases, $DB_{orig}$, $DB_{NeNDS}$, $DB_{geom}$, and $DB_{GT-NeNDS}$ are. The number of clusters for K-means is specified as an input to the algorithm.

The evaluation of the usability is performed in three parts. In the first part, a qualitative analysis of the clustering algorithm is carried out by comparing the performance of the K-means algorithm on the original database, the database obfuscated using geometric transformations, and on the database obfuscated using NeNDS. In the second part, a quantitative analysis of the distortions produced by the data obfuscation technique is evaluated based on the Misclassification Error percentage. The performance of both hierarchical clustering and K-means on data obfuscated by NeNDS is compared with the original results for

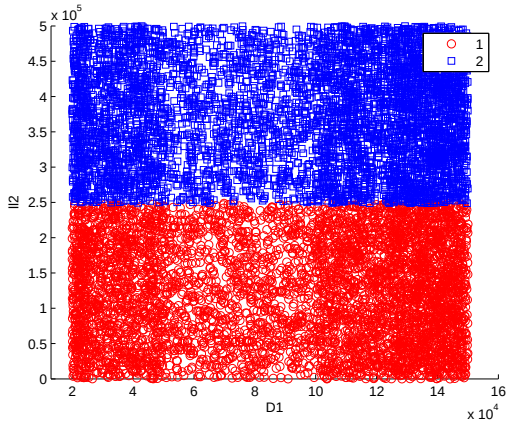
different neighborhood sizes. The third part evaluates the effect the neighborhood size of NeNDS has on the misclassification error percentage and the computation time.

### 6.2.3 Qualitative Analysis

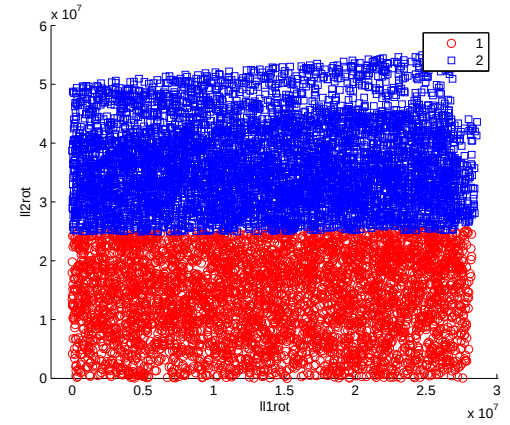
The preliminary evaluation of the cluster preserving property of NeNDS and GT-NeNDS is performed here. In this set of experiments, the neighborhood size is fixed and equal to the total number of records in the database. The aim of the experiments is to assess the feasibility of using NeNDS transformed data for data mining. A qualitative analysis is performed in this set of experiments and the results are plotted on a 2-D graph.

The clustering algorithms can produce clusters on N-dimensional data. However, for the ease of visualizing the results, clustering is performed only on pairs of attributes in this section. Three attributes of the synthetic database, **Salary, Commission, and Age**, are obfuscated using NeNDS and rotation-based geometric-transformation. The three attributes are denoted as $D_1, D_2, D_3$ in the graphs. The K-means algorithm is then applied to each pair of attributes of each of the databases $DB_{orig}$, $DB_{NeNDS}$, and $DB_{geom}$, and the average of a 100 runs of K-means are plotted on a graph. Each set of graphs represents a different value of $K$ or number of clusters for the K-means algorithm.

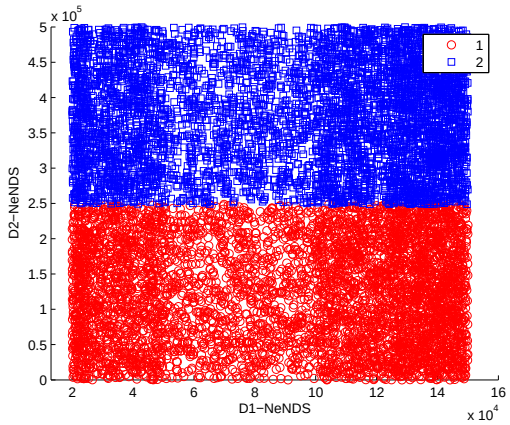
Figures 7, 8, 9, 10, 11, and 12 show the K-means clustering results for the synthetic database. In the graphs in Figures 7, and 8, the databases are generated with an inherent cluster degree $C_{in} = 2$ and $10,000$ records. Rotation is performed with respect to the pairs $(D1, D2)$ and $(D1, D3)$. The angle of rotation used for geometric transformation is $89.4^o$. The different colors indicate the clusters to which the data points belong. The color codes in each graph are independent and only identify the different clusters in the graph. In Figure 7, the K-means algorithm is run with $K = 2$. The algorithm divides the data into two clusters. It is observed that two distinct data clusters are created for all three databases $DB_{orig}$, $DB_{NeNDS}$, and $DB_{geom}$. This shows that for a small number of clusters, NeNDS and geometric transformations do not distort the results of clustering. The results of running K-means with $K = 5$ is shown in Figure 8. The results in the figure show that in this case, the obfuscated databases perform as well as the original database even when




(a) Original Data




(b) Rotated Data




(c) NeNDS transformed Data

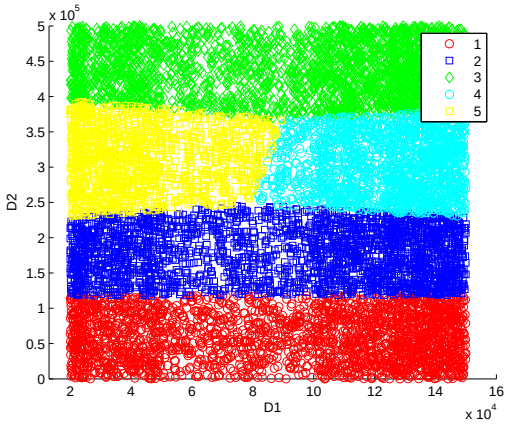
**Figure 7:** Comparison of DO techniques for Large Database ($N = 10,000, C_{in} = 2, K = 2$).

the number of clusters $K$ is greater than the inherent clustering degree $C_{in}$.

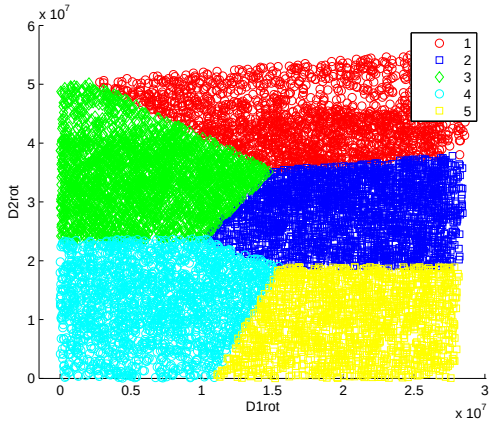
The experiment results shown in Figures 9 and 10 are obtained by running a K-means algorithm with $K = 5$ on a synthetic database generated with an inherent clustering degree $C_{in} = 5$ and $2,125$ records. The angle of rotation between $D1$ and $D2$ is $89.9^o$. Figure 9 shows the results for $D1$ versus $D2$ for the original, rotated, and NeNDS transformed databases. The clusters appear to be identical in all three cases. The graphs in Figure 10 show the results of the K-means algorithm with $K = 5$ plotted for $D2$ versus $D3$. The angle of rotation is changed to 35.4 for this experiment. The rotation graph appears to be significantly different from the other two. The reason for this is the angle of rotation. The proof of cluster preservation is provided in Section 6.2.3.

The experiment results shown in Figure 11 and 12 use the same database as Figures 9 and 10. The angle of rotation between $D1$ and $D2$ is $89.9^o$. The results of clustering $D1$ and $D2$ using K-means with $K = 10$ are shown in Figure 11. Some distortions in the clusters can be observed in the NeNDS transformed data. Figure 12 shows similar results, indicating that when the number of clusters is much larger than the inherent clustering degree $C_{in}$ of the database, the results are distorted slightly. The amount of distortion introduced is evaluated in Section 6.2.3.

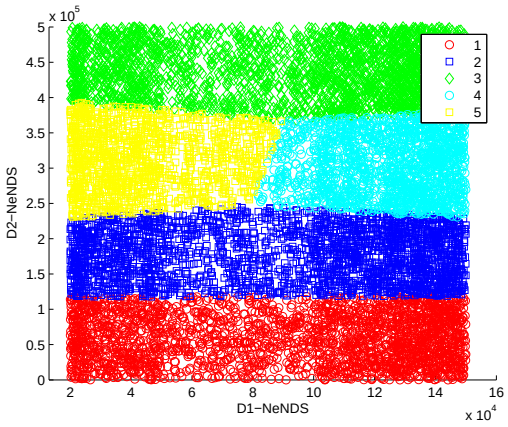
The results of clustering data obfuscated using GT-NeNDS are shown in Figures 13 and 14. Both the experiments are performed on real data. The databases are clustered using K-means clustering with $K = 5$. The database used in Figure 13 uses two attributes from the diabetes dataset that has 150 records. A scaling factor of 0.6 is used for scaled-NeNDS and an angle of rotation $90^o$ is used for rotated-NeNDS. It is observed that the shape of the graph is similar in all three cases showing that clustering is preserved. The thyroid database used in Figure 14 contains $7,00$ records. Only two of the attributes are used in this experiment. The angle of rotation used for GT-NeNDS is $45^o$. The shapes of the clusters appear to have changed in this case. A more quantitative evaluation of the performance of NeNDS and GT-NeNDS is provided in the next section.




(a) Original Data

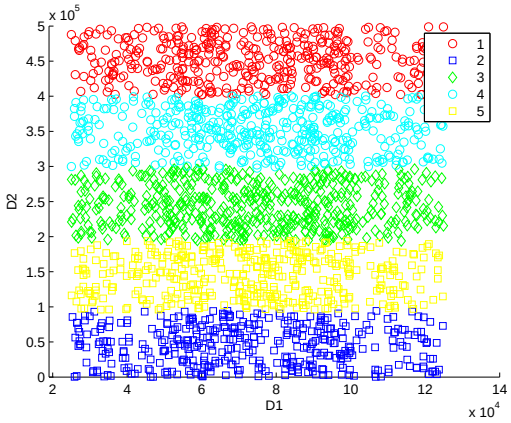



(b) Rotated Data

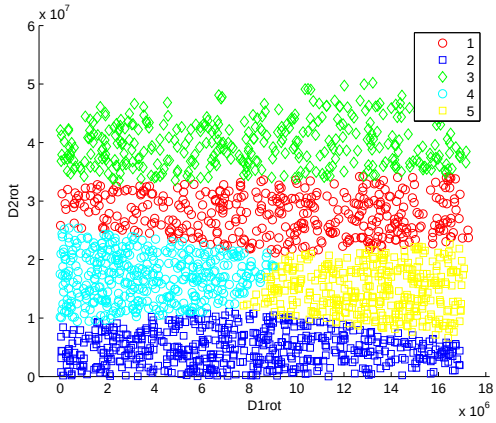



(c) NeNDS transformed Data

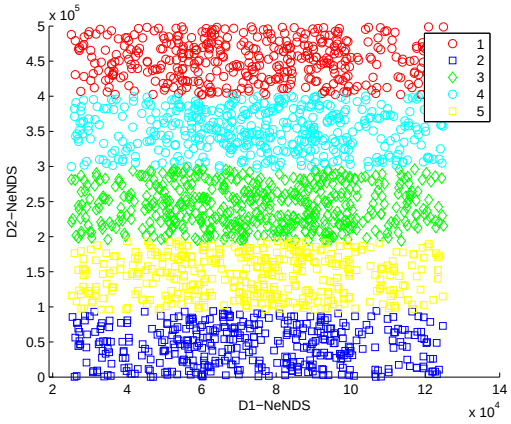
**Figure 8:** Comparison of DO techniques for large database for ($N = 10,000, Cin = 2, K = 5$).




(a) Original Data

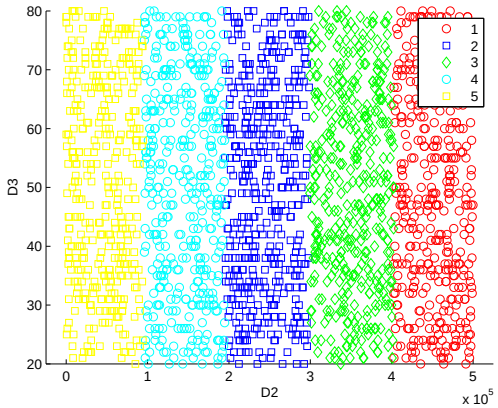



(b) Rotated Data

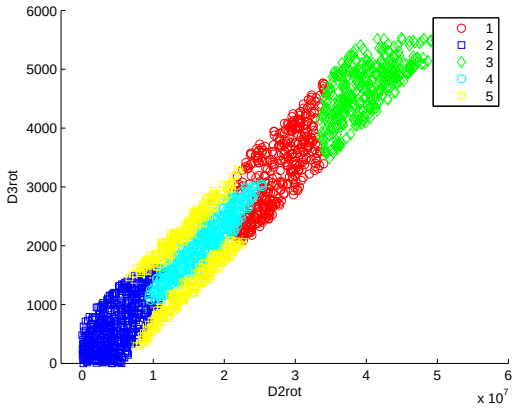



(c) NeNDS transformed Data

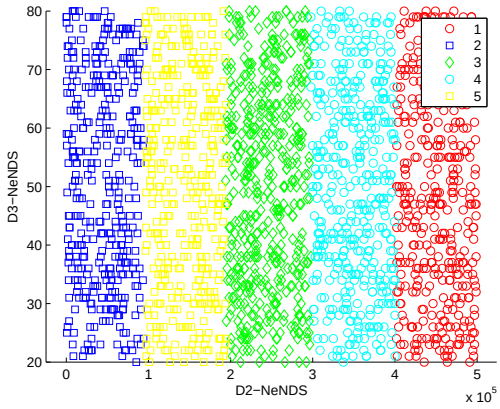
**Figure 9:** Comparison of DO techniques, for (D1,D2), $(N = 2,125, C_{in} = 5, K = 5)$.




(a) Original Data

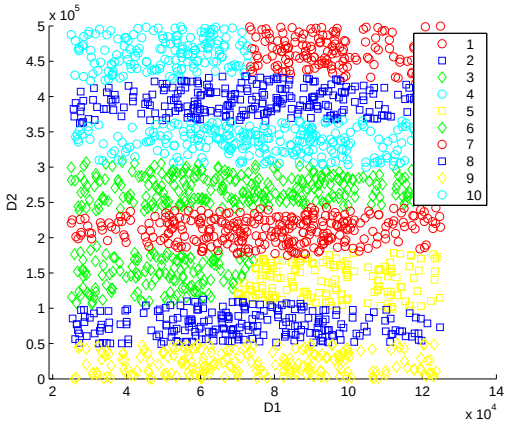



(b) Rotated Data

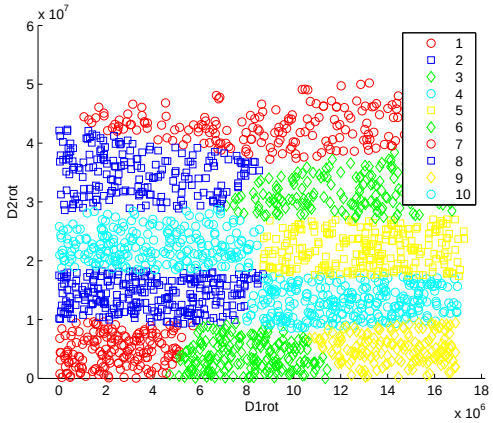



(c) NeNDS transformed Data

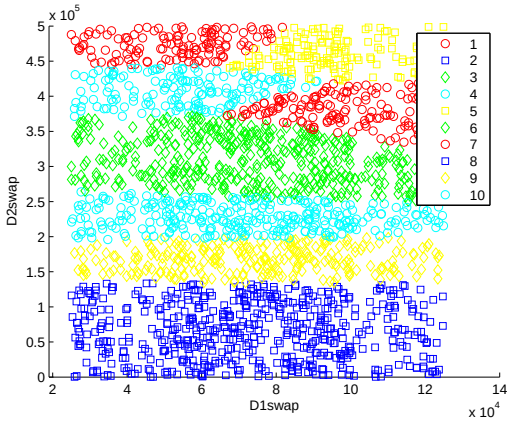
**Figure 10:** Comparison of DO techniques, for (D2,D3), $(N = 2,125, C_{in} = 5, K = 5)$.




(a) Original Data

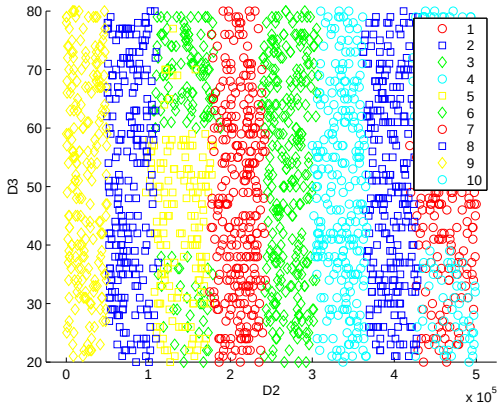



(b) Rotated Data

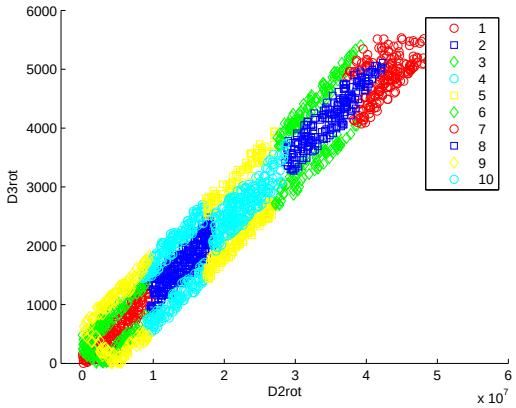



(c) NeNDS transformed Data

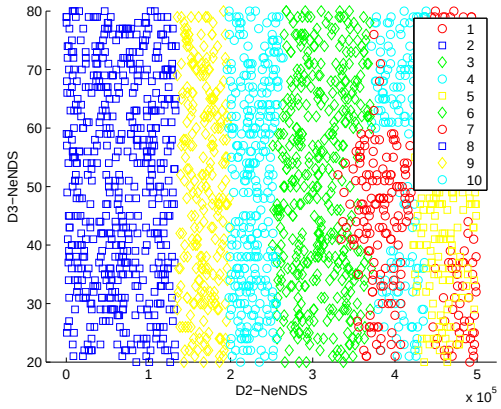
**Figure 11:** Comparison of DO techniques, for (D1,D2), $N = 2,125, C_{in} = 5, K = 10$.




(a) Original Data

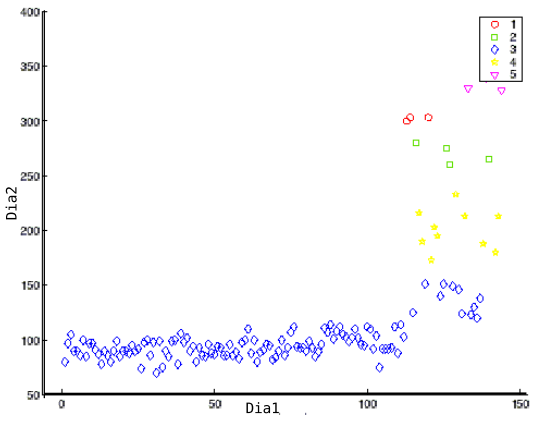



(b) Rotated Data

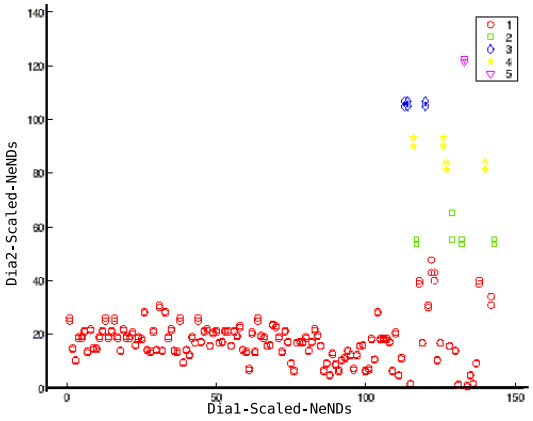



(c) NeNDS transformed Data

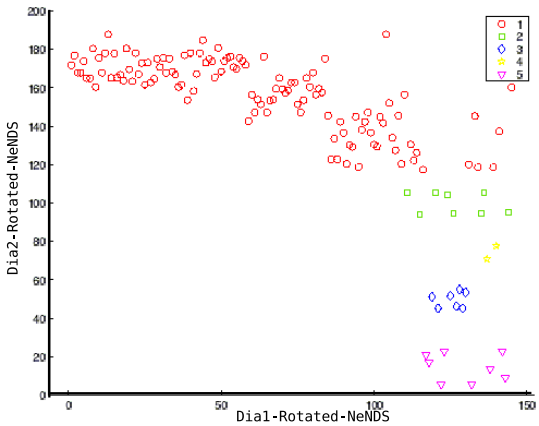
**Figure 12:** Comparison of DO techniques, for (D2,D3), $N = 2,125, C_{in} = 5, K = 10$.




(a) Original Data

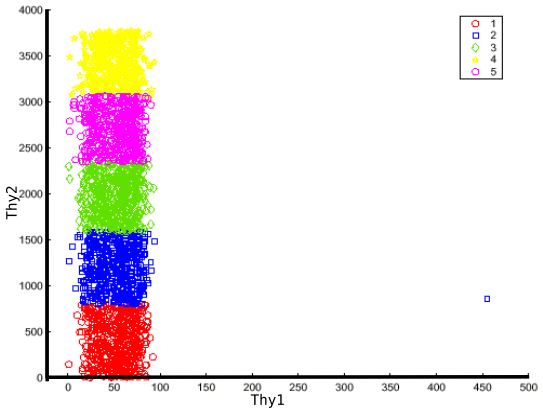



(b) Scaled-NeNDS Data

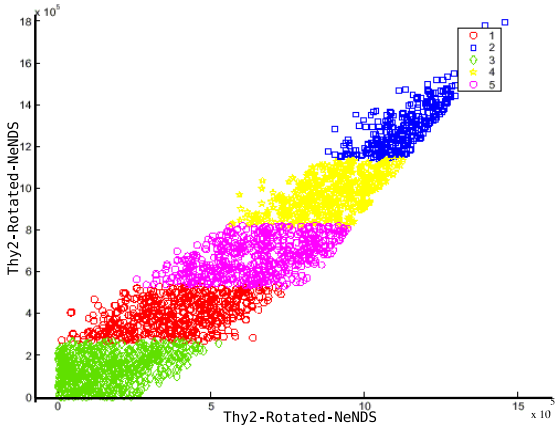



(c) Rotated-NeNDS Data

**Figure 13:** Performance of GT-NeNDS using the Diabetes database with N = 150.




(a) Original Data




(b) Rotated-NeNDS Data

**Figure 14:** Performance of GT-NeNDS using the Thyroid database with N = 7,200.

### 6.2.4 Quantitative Analysis

A qualitative assessment of NeNDS and GT-NeNDS was performed in the previous section. A more comprehensive and quantitative analysis of the distortions in clustering due to NeNDS-based data obfuscation is performed here. A measure known as the *Misclassification Error Percentage* is used to compute the distortion produced by data obfuscation. The metric was proposed in [63] to evaluate the number of data points that have moved from one cluster to another. The average number of clusters that have moved from their original clusters is computed using Equation 25, where $N$ is the total number of records in the data set, $X : X \in D_{k,n}$ represents a data item with $n$ fields, $K$ is the number of clusters into which the data are grouped, and $Cluster_i(X)$ is the original cluster and $Cluster_i(X')$ is the new cluster formed from the obfuscated data.

$$MCE = \frac{1}{N} * \sum_{i=1}^{K} (|Cluster_i(X)| - |Cluster_i(X')|) \tag{25}$$

This section also evaluates the effect of the clustering results when different neighborhood sizes are used for NeNDS and GT-NeNDS. The tests are carried out for minimum neighborhood sizes $NH_{size}$ varying from 1% of the database to 20% of the database for real and synthetic data. The different neighborhood sizes for each database are listed in Table 19. The actual size of some of the neighborhoods in the may be larger than the minimum size because of one of two reasons

- The existence of duplicate entries in a dataset could potentially lead to the creation of one or more neighborhoods with size greater than $NH_{size}$. For these experiments, the threshold for permuting duplicates is set to be equal to the value of the minimum neighborhood size parameter $NH_{size}$. If the number of duplicates is less than $NH_{size}$, the duplicates are included in the permutation process. This is done by merging up to 3 closest neighborhoods into one. For instance, let a data set has 1000 items and the specified value for $NH_{size} = 30$ If the number of duplicates $N_{dup_i}$ of any item $i$ is less than or equal to 10, then a neighborhood of size $NH_{size}$ is sufficient to permute the duplicate elements uniquely. If $10 < N_{dup_i} < 20$, then the minimum size of the

neighborhood required for permuting each duplicate entry uniquely is $3 * N_{dup_i} = 60$ items. The neighborhood containing the duplicate items is merged with its closest neighbor to form a neighborhood of size 60. For values of $10 < N_{dup_i} < 30$, the minimum size of the neighborhood required is 90, which is three times $NH_{size_{min}}$. In this case, the neighborhood containing the duplicate items is merged with two of its closest neighbors. For all items $i$ for which $N_{dup_i} > NH_{size}$, the duplicate items are retained without permuting. The neighborhood size sets a bound on the permissible level of anonymity to the set. In this case, even if the attacker is aware of the exact value of the data item, presence of $N_{dup_i}$ instances protects the identity of the record. The maximum size of any neighborhood in a data set is limited to three times the specified value $NH_{size}$.

- If the minimum neighborhood size $NH_{size}$ specified is not a factor of the total size of the data set, one of the neighborhoods is likely to have fewer than the specified number of elements. This situation is also likely in the presence of duplicate items in the data set. In these cases, the neighborhood is merged with a neighborhood closest to it.

The maximum number of neighborhoods into which the data sets are partitioned are $NH = \lfloor \frac{N}{NH_{size}} \rfloor$, where $N$ is the number of items in the data set. The maximum number of neighborhoods for each of the databases specified in Table 19 are $[100, 50, 20, 10, 5]$ corresponding to the different neighborhood sizes $[1\%, 2\%, 5\%, 10\%, 20\%]$ of $N$ respectively. Each data set of the database is divided into a maximum of $NH$ neighborhoods. NeNDS is applied to each neighborhood of each data set to produce $NH$ obfuscated neighborhoods for every data set.

The performance of hierarchical and K-means clustering algorithms are evaluated in this section. Table 20 shows the results of applying K-means with $K = 5$ on a synthetic database generated with an inherent clustering degree $C_{in} = 10$, containing $3,000$ records. The six non-categorical attributes are obfuscated using NeNDS with varying neighborhood sizes ($NH_{size} = [1\%, 2\%, 5\%, 10\%, 20\%]$). The results are averaged over 100 runs of the K-means

**Table 19:** Neighborhood sizes.

| Database | 1 % | 2 % | 5 % | 10 % | 20 % |
|---|---|---|---|---|---|
| Synthetic (3000) | 30 | 60 | 150 | 300 | 600 |
| Synthetic (5000) | 50 | 100 | 250 | 500 | 1000 |
| Thyroid (7200) | 72 | 144 | 360 | 720 | 1440 |

algorithm. The six non-categorical fields in the database are obfuscated using NeNDS for different neighborhood sizes to yield 5 NeNDS obfuscated database $DB_{NeNDS_i}, i \in [1, 5]$. For geometric transformation, a rotation vector is generated each time, which is multiplied to the original database to produce one database $DB_{geom}$. The datasets $DB_{NeNDS_i}$ are geometrically transformed by multiplying the databases by a randomly generated rotation vector to produce 5 GT-NeNDS obfuscated databases $DB_{GT-NeNDS_i}$. The three columns in the table represent the $MCE\%$ resulting from clustering data obfuscated using geometric transformation (rotation), NeNDS, and GT-NeNDS respectively. Since geometric transformations are independent of the number of neighborhoods, all the entries in the first column are identical in the table. A 0.04% misclassification error for geometric transformation implies that in the database containing 3000 records and 6 fields, an average of one record is displaced to a different cluster as a result of the geometric transformation. For NeNDS-transformed data, the results are similar and show that an average of 1 record moves from one cluster to another as a result of the transformation. The displacement of clusters for GT-NeNDS is similar to NeNDS, indicating that all three obfuscation techniques perform similarly for this set of experiments. There is a slight change in the $MCE$ percentage for different neighborhood sizes. However, the actual number of records displaced is still the same.

In Table 21, K-means clustering is performed on the same databases as the experiment above. The only parameter changed here is the number of clusters $K$ that is specified for creating clusters. For this set of experiments, K-means clustering is performed with $K = 10$, which is equal to the inherent degree of clustering $C_{in}$ of the generated database. The $MCE\%$ for all the obfuscated databases are higher than the values obtained for $K = 5$.

**Table 20:** Clustering versus neighborhood size using K-Means, K=5, N = 3000.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 30 | 0.04 | 0.03 | 0.03 |
| 60 | 0.04 | 0.03 | 0.03 |
| 150 | 0.04 | 0.03 | 0.04 |
| 300 | 0.04 | 0.04 | 0.04 |
| 600 | 0.04 | 0.04 | 0.04 |

For the geometrically transformed database, the $MCE\%$ is increased to 0.13%. For larger number of neighborhoods NeNDS is observed to perform marginally better than geometric transformations. The performance of GT-NeNDS is slightly worse than the other two algorithms. However, the actual change in the number of records that are displaced is less than 1 record in the 3000 record database. The average number of records that are displaced for all the databases is 4.

**Table 21:** Clustering versus neighborhood size using K-Means, K=10, N = 3000.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 30 | 0.13 | 0.12 | 0.13 |
| 60 | 0.13 | 0.12 | 0.13 |
| 150 | 0.13 | 0.13 | 0.14 |
| 300 | 0.13 | 0.14 | 0.14 |
| 600 | 0.13 | 0.14 | 0.14 |

The effect on the clusters when the number of clusters for K-means exceeds the clustering degree $C_{in}$ is evaluated and shown in Table 22. When the number of clusters $K$ exceeds the degree of clustering in the database, the clusters produced by the K-means algorithm are not optimal. This is because some of the resulting clusters are likely to be very different from the inherent clustering of the data. Algorithms for predicting the correct number of clusters have been proposed and implemented due to this drawback of K-means. In this dissertation, the aim of the experiments is to study the distortion in the results of clustering, which includes the distortions produced for the correct number of clusters as

well as distortions produced when the number of clusters are less or more than the exact number of clusters. The value of $K$ for this set of experiments is 15, which implies that the K-means algorithm creates 15 clusters from the database that has a clustering degree of 10. The results in Table 22 show that the $MCE\%$ for geometrically transformed databases is 0.27. This is almost twice the $MCE\%$ for K-means with $K = 10$. The distortions result from the less than optimal selection of $K$. NeNDS has an average $MCE\% = 0.27$. The difference between the smallest and largest $MCE\%$ is 0.04, which indicates that the choice of different values for $NH$ changes the clustering results by a maximum of 1 record. The average $MCE\%$ for GT-NeNDS ranges from $0.27 - 0.31$, which is similar to NeNDS. It is observed that the change in $MCE\%$ are almost negligible for different number of neighborhoods.

**Table 22:** Clustering versus neighborhood size using K-Means, K=15, N = 3000.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 30 | 0.27 | 0.27 | 0.27 |
| 60 | 0.27 | 0.26 | 0.29 |
| 150 | 0.27 | 0.29 | 0.31 |
| 300 | 0.27 | 0.30 | 0.30 |
| 600 | 0.27 | 0.29 | 0.31 |

The results indicate that for this set of experiments the distortions produced by obfuscating the data are very small when the data are clustered using K-means clustering irrespective of the size of the neighborhoods.

Tables 23, 24 and 25 show the effect of obfuscated data on the performance of hierarchical clustering algorithms. The three sets of experiments carried out for K-means clustering are repeated for evaluating the distortions produced when hierarchical clustering is used. The GAVE algorithm is used for hierarchical clustering and the resulting tree of clusters are cut at the rows corresponding to $K = 5$, $K = 10$ and $K = 15$. The results of the hierarchical clustering tree corresponding to $K = 5$ are shown in Table 23. For all the databases, obfuscated using geometric transformation, NeNDS and GT-NeNDS, the average $MCE\%$

for all the databases is 0.08. The $MCE\%$ for the geometrically transformed database is 0.06%. For NeNDS, the error percentage varies between $0.06-0.08$. The changes in $MCE\%$ with respect to the number of neighborhoods is random and the actual variation produced is extremely small. GT-NeNDS performs similar to NeNDS with the only change being a slightly higher $MCE\%$.

**Table 23:** Clustering versus neighborhood size using Hierarchical Clustering,$K = 5$, $N = 3000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 30 | 0.06 | 0.07 | 0.08 |
| 60 | 0.06 | 0.06 | 0.07 |
| 150 | 0.06 | 0.07 | 0.09 |
| 300 | 0.06 | 0.08 | 0.0.08 |
| 600 | 0.06 | 0.07 | 0.0.08 |

Table 24 shows the results of the hierarchical cluster tree for $K = 10$. This is also equal to the inherent clustering degree of the database. The results obtained for hierarchical clustering are similar to the results produced by applying the K-means clustering algorithm on the databases as shown in Table 21. Here again, NeNDS and GT-NeNDS transformed databases perform slightly worse than the geometrically transformed databases. In one of the experiments, with 150 records, NeNDS performs marginally better than the geometrically transformed database.

**Table 24:** Clustering versus Neighborhood size using Hierarchical Clustering, $K = 10$, $N = 3000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 30 | 0.14 | 0.12 | 0.14 |
| 60 | 0.14 | 0.14 | 0.14 |
| 150 | 0.14 | 0.13 | 0.14 |
| 300 | 0.04 | 0.14 | 0.15 |
| 600 | 0.04 | 0.15 | 0.15 |

In Table 25 the hierarchical cluster tree is cut at the row corresponding to $K = 15$. The

number of clusters created is greater than the inherent clustering degree of the generated database. Hierarchical clustering behaves similar to K-means when the number of clusters exceeds the optimum clustering degree of the database. The results of this set of experiments shows that the distortions produced when the number of clusters created is greater than the inherent clustering are higher than the distortions resulting from clustering the data with $K$ being less than or equal to the clustering degree. The average distortion for the geometrically transformed databases is $MCE\% = 0.28$. As was the case with K-means, the results of the hierarchical clustering on the obfuscated databases indicates that the $MCE\%$ with $K = 15$ is almost twice the value for $K = 10$. The *Elbow* criterion for the GAVE algorithm is found to be $9 - 11$. This shows that the optimum number of clusters were formed for $K = [9, 11]$.

**Table 25:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 15$, $N = 3000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 30 | 0.28 | 0.26 | 0.29 |
| 60 | 0.28 | 0.27 | 0.29 |
| 150 | 0.28 | 0.28 | 0.31 |
| 300 | 0.28 | 0.30 | 0.34 |
| 600 | 0.28 | 0.30 | 0.31 |

Tables 26, 27, and 22 show the effect of the neighborhood sizes on clustering for a larger database $N = 5,000$. The inherent clustering degree for this data is $C_{in} = 20$. The performance of K-means and hierarchical for $K = [10, 15, 20]$ are shown. For $K = 10$, the results in Table 26 show that the MCE percentage for geometrically transformed data is 0.11%. For this database, the error corresponds to a displacement of 5 records on an average. For NeNDS and GT-NeNDS the average value of the distortions is less than one in all but one case, where the average distortion corresponds to $5 - 7$ records being displaced from their original clusters.

The results in Table 27 are obtained by applying K-means clustering to the databases with $K = 15$. The values of $K$ is still less than the inherent clustering degree. The

**Table 26:** Clustering versus neighborhood size using K-Means, $K = 10$, $N = 5,000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 50 | 0.11 | 0.11 | 0.12 |
| 100 | 0.11 | 0.12 | 0.15 |
| 250 | 0.11 | 0.11 | 0.12 |
| 500 | 0.11 | 0.13 | 0.14 |
| 1000 | 0.11 | 0.14 | 0.14 |

results here indicate that on an average, one record is displaced from its original cluster in a database with $5,000$ records. Similar results are obtained when the value of $K$ is increased to 15. The average $MCE\%$ for these databases is $MCE\% = 0.26$. The average performance of the geometrically transformed database and NeNDS-transformed databases are slightly lower than the average. However, NeNDS performs marginally better than geometric transformations for $NH = 50, 100$ neighborhoods. The performance of GT-NeNDS is marginally higher than the average.

**Table 27:** Clustering versus neighborhood size using K-Means, $K = 15$, $N = 5,000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 50 | 0.25 | 0.24 | 0.26 |
| 100 | 0.25 | 0.23 | 0.27 |
| 250 | 0.25 | 0.26 | 0.26 |
| 500 | 0.25 | 0.26 | 0.28 |
| 1000 | 0.25 | 0.27 | 0.28 |

**Table 28:** Clustering versus neighborhood size using K-Means, $K = 20$, $N = 5,000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 50 | 0.32 | 0.30 | 0.32 |
| 100 | 0.32 | 0.31 | 0.33 |
| 250 | 0.32 | 0.31 | 0.34 |
| 500 | 0.32 | 0.35 | 0.34 |
| 1000 | 0.32 | 0.34 | 0.35 |

Tables 29, 30, and 31 show how the $MCE\%$ varies when hierarchical clustering is applied to NeNDS transformed data, geometrically transformed data, and GT-NeNDS. The cluster tree resulting from GAVE is cut at the appropriate rows corresponding to $K = [10, 15, 20]$. The results obtained for hierarchical clustering have the same trends as K-means. In all three cases, NeNDS performs slightly better than geometric transformations for $NH = 100, 250$, but the average value of $MCE\%$ is equal to or slightly greater than the $MCE\%$ for geometric transformations. The performance of GT-NeNDS is marginally worse than NeNDS and geometric transformations.

**Table 29:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 10$, $N = 5,000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 50 | 0.11 | 0.13 | 0.13 |
| 100 | 0.11 | 0.12 | 0.13 |
| 250 | 0.11 | 0.12 | 0.13 |
| 500 | 0.11 | 0.13 | 0.14 |
| 1000 | 0.11 | 0.13 | 0.14 |

**Table 30:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 15$, $N = 5,000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 50 | 0.24 | 0.26 | 0.27 |
| 100 | 0.24 | 0.25 | 0.25 |
| 250 | 0.24 | 0.24 | 0.26 |
| 500 | 0.24 | 0.23 | 0.26 |
| 1000 | 0.24 | 0.25 | 0.27 |

In Table 31 it is seen that for geometrically transformed databases, the $MCE\% = 0.36$ indicating that $15 - 16$ records are displaced from their original clusters as a result of data obfuscation. For NeNDS and GT-NeNDS, the $MCE\% = 0.35 - 0.39$. In the best case, for $NH = 100$, 250, the performance of NeNDS is marginally better than geometric transformations. The difference between the maximum $MCE\% = 0.39$ and the average

**Table 31:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 15$, $N = 5,000$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 50 | 0.36 | 0.34 | 0.39 |
| 100 | 0.36 | 0.35 | 0.39 |
| 250 | 0.36 | 0.34 | 0.38 |
| 500 | 0.36 | 0.37 | 0.39 |
| 1000 | 0.36 | 0.36 | 0.37 |

value $MCE\% = 0.37$ is 0.02, which corresponds to a difference of 1 record in the database

The results obtained when clustering algorithms K-means and hierarchical clustering are evaluated in the following sets of experiments. The Thyroid database with 21 fields and $7,200$ records is used for evaluation. In Table 32, the obfuscated databases are clustered using K-means clustering with $K = 20$. The inherent clustering degree for this database is not known as it is a real database. The results show that for all the obfuscated databases the $MCE\%$ is 0.32. The geometrically transformed database results in an $MCE\%$ that is slightly less than the average distortion. In this experiment, both NeNDS and GT-NeNDS perform slightly worse than geometric transformation.

**Table 32:** Clustering versus neighborhood size using K-Means, $K = 20$, $N = 7,200$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 72 | 0.28 | 0.30 | 0.31 |
| 144 | 0.28 | 0.30 | 0.32 |
| 360 | 0.28 | 0.30 | 0.32 |
| 720 | 0.28 | 0.32 | 0.33 |
| 1440 | 0.28 | 0.32 | 0.34 |

Table 33 shows the results of applying K-means clustering to the obfuscated databases with $K = 30$. The results indicate MCE of $[0.45\%, 0.49\%, 0.51\%]$ for the obfuscated databases. The increase in $MCE\%$ is attributed to the increase in the number of clusters K for K-means from 20 to 30. Since the database used in this experiment is a real

database, the inherent clustering of the database is not known *a priori*. However, the results seem to indicate that the value of $K$ is larger than the inherent clustering degree.

**Table 33:** Clustering versus neighborhood size using K-Means, $K = 30$, $N = 7,200$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 72 | 0.45 | 0.49 | 0.48 |
| 144 | 0.45 | 0.49 | 0.49 |
| 360 | 0.45 | 0.48 | 0.49 |
| 720 | 0.45 | 0.5 | 0.51 |
| 1440 | 0.45 | 0.5 | 0.54 |

Table 34 shows the results of applying K-means to the obfuscated databases with $K = 40$. The results obtained in this case are similar to the results in Table 33. The average $MCE\%$ in this case is 0.85. The change in number of neighborhoods changes the $MCE\%$ only by a small amount. The worsening of the $MCE\%$ for $K = 40$ strengthens the guess that the number of clusters for K-means clustering is larger than the inherent clustering of the database.

**Table 34:** Clustering versus neighborhood size using K-Means, $K = 40$, $N = 7,200$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 72 | 0.82 | 0.83 | 0.86 |
| 144 | 0.82 | 0.83 | 0.90 |
| 360 | 0.82 | 0.85 | 0.91 |
| 720 | 0.82 | 0.91 | 0.91 |
| 1440 | 0.82 | 0.90 | 0.91 |

Tables 35, 36, and 37 show the results of performing hierarchical clustering on the Thyroid database containing $7,200$ records and 31 fields. The three tables show the results of the GAVE clustering tree cut at the rows corresponding to $K = 20$, $K = 30$, and $K = 40$. The *Elbow* for GAVE is found to be between $20 - 22$ clusters. The results obtained for hierarchical clustering resembles the performance for K-means clustering very closely. In each case, the results for the geometrically transformed database are marginally better

than NeNDS and GT-NeNDS. The actual difference between the $MCE\%$ for the best and worst cases is less than 0.1 for all cases. The number of neighborhoods does not have a significant impact on the performance of the clustering algorithms.

**Table 35:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 20$, $N = 7,200$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 72 | 0.31 | 0.33 | 0.35 |
| 144 | 0.31 | 0.32 | 0.35 |
| 360 | 0.31 | 0.33 | 0.34 |
| 720 | 0.31 | 0.34 | 0.36 |
| 1440 | 0.31 | 0.33 | 0.34 |

**Table 36:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 30$, $N = 7,200$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 72 | 0.47 | 0.46 | 0.48 |
| 144 | 0.47 | 0.49 | 0.49 |
| 360 | 0.47 | 0.48 | 0.49 |
| 720 | 0.47 | 0.49 | 0.50 |
| 1440 | 0.47 | 0.49 | 0.50 |

The results indicate that the performance of geometric transformations, NeNDS, and GT-NeNDS are extremely close to the results without obfuscation when the number of clusters created is close to the optimal value. There is a slight worsening of the performance when the granularity is increased. However, as seen from the tables, the error percentage even in the worst case is less than 1% of the database.

### 6.2.5 Neighborhood Size and Time Complexity

The distortions produced when clustering algorithms are applied to obfuscated data have been evaluated for two types of clustering techniques- K-means and hierarchical clustering. The results of the experiments show that NeNDS and GT-NeNDS produce very small

**Table 37:** Clustering versus neighborhood size using Hierarchical Clustering, $K = 40$, $N = 7,200$.

| Neighborhood Size | Rotation | NeNDS | GT-NeNDS |
|---|---|---|---|
| 72 | 0.81 | 0.83 | 0.85 |
| 144 | 0.81 | 0.83 | 0.87 |
| 360 | 0.81 | 0.85 | 0.88 |
| 720 | 0.81 | 0.87 | 0.89 |
| 1440 | 0.81 | 0.84 | 0.84 |

distortions to the inherent clustering of the databases. Both types of clustering technique produced similar results for all the experiments. The effect of varying the neighborhood size for NeNDS analyzed for two types of clustering algorithms and different neighborhood sizes. Clustering experiments were carried out for different neighborhood sizes to evaluate the effect of different neighborhood sizes on the creation of clusters. The experiments showed that the distortions introduced by changing the neighborhood sizes are very small.

Figure 15 shows a graph of the $MCE\%$ versus the number of neighborhoods, where the number of neighborhoods is increased from $0-100$ for the synthetic database generated with an inherent clustering $C_{in} = 10$ and $3,000$ records. The number of neighborhoods $NH$ is expressed as $\lfloor \frac{N}{NH_{size}} \rfloor$, where $NH_{size}$ is the neighborhood size and $N$ is the size of the data set. The graph shows the $MCE\%$ varies only slightly when the number of neighborhoods are increased from $0 - 100$. The difference between the extreme values of $MCE\%$ is $0.02\%$ indicating that by changing the neighborhood size from $0 - 100$ alters less than 1 record in the database. Plotting the $MCE\%$ versus number of neighborhoods for different databases produced similar results. This shows that the choice of neighborhood size $NH_{size}$ or the number of neighborhoods $NH$ has little or no effect on the misclassification error.

The size of the neighborhood and the number of neighborhoods for NeNDS does not affect the $MCE\%$ significantly. However, a large neighborhood is likely to take longer time to compute the permutation candidate than a smaller neighborhood. Figure 16 shows a graph of the Computation time ($t$) versus the number of neighborhoods for a synthetic database containing $3,000$ records and an inherent clustering degree $C_{in} = 10$. The graphs show

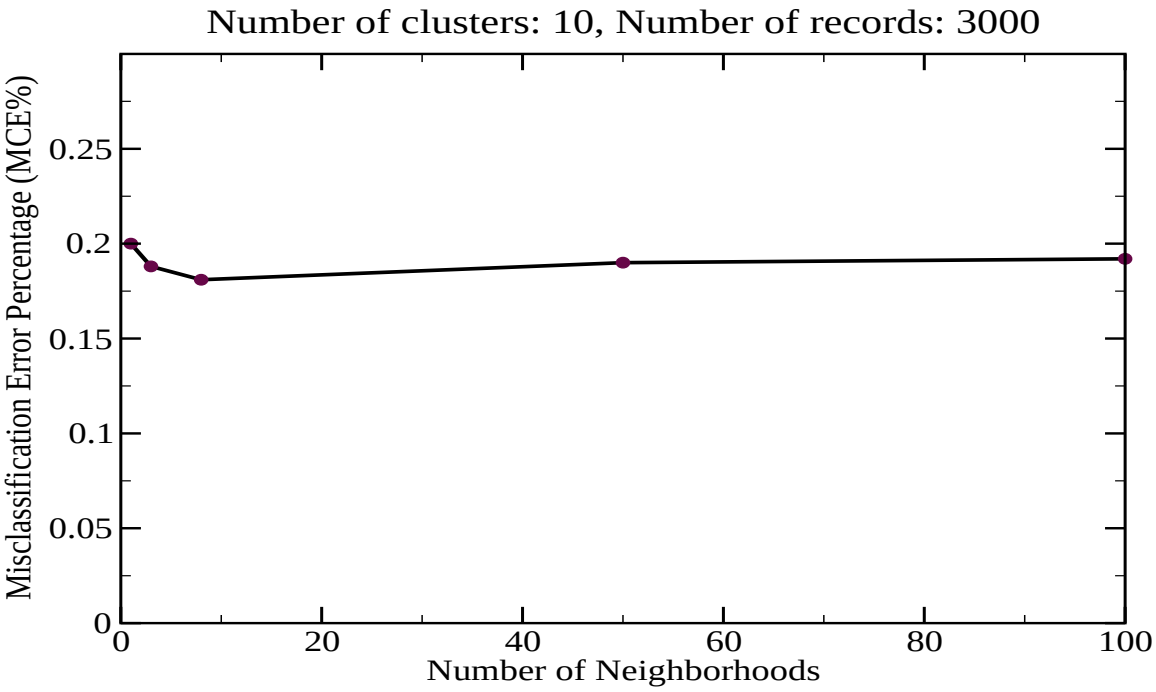



**Figure 15:** Effect of misclassification error on neighborhood size.

that the computation time decreases exponentially when the number of neighborhoods are increased. When the neighborhood size $NH = 1$, each data set (field) in the database is treated as a single neighborhood with $3,000$ data items. Increasing the number of neighborhoods decreases the size of each neighborhood. This results in reducing the size of the tree for computing the optimum permutation candidate. Although the graph is exponential in nature, the computation time for the worst case ($NH = 1$) is a finite number and was completed in $1,700$ seconds. This is enabled by the heuristic branch and bound tree traversal algorithm for NeNDS. It is seen that by increasing the number of neighborhoods by a small number, a significant reduction in computation time is obtained. Choosing a neighborhood size up to $1/3$ of the entire data set will still result in a reasonably fast obfuscation that provides good privacy and preserves clustering.

### 6.2.6 Clustering Randomized Data

The distortion of clusters produced by clustering data obfuscated using random data perturbation is evaluated to assess the usability of randomized data. The RDP algorithm takes as

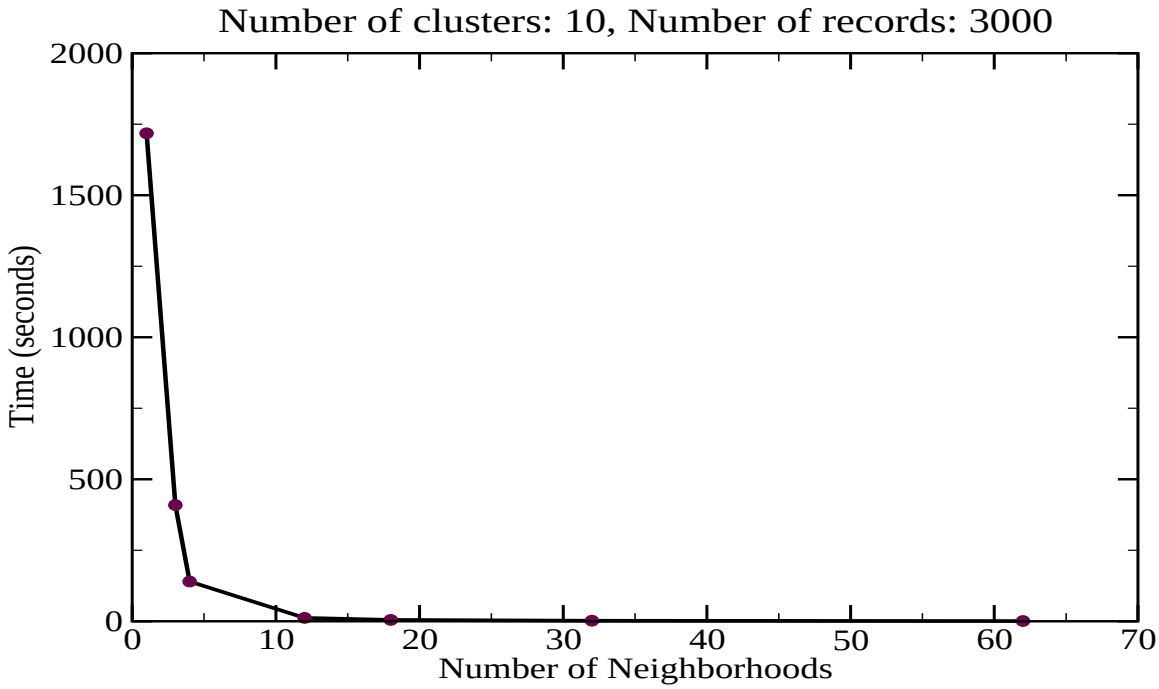



**Figure 16:** Effect of computation time on neighborhood size.

input the mean and standard deviation $[\mu, \sigma]$ and generates a Gaussian distribution, which is then added to the original data set. Following this, a second Gaussian distribution with the same parameters is generated and subtracted from the noise added data set. The final database consists of the datasets to which a noise distribution is added and a second distribution with the same parameters is subtracted. Four randomized databases are generated, each with a different set of parameters for the noise distribution. The parameters chosen are $[0, 1], [0, 10], [0, 20], [1, 5]$. The first parameter produces a noise distribution with 0 mean and a standard deviation of 1. The resulting randomized data sets by a very small value. The second and third parameter sets generate noise distributions that have 0 mean but larger deviations. The last parameter generates a skewed distribution because of the non-zero mean. The experiments are carried out for the two synthetic databases, $N = [3000, 5000]$ and the real database with $7,200$ records of Thyroid data.

Table 38 and 39 show the $MCE\%$ results when the randomized databases are clustered using the K-means clustering and hierarchical clustering algorithms respectively for $K = [10, 20, 30]$. The misclassification error is very small $0.01\%$ for randomized data with a

small noise distribution [0.1]. For all other cases, the $MCE\%$ is much larger range from $13.5\% - 45.6\%$. The misclassification error is worse when the number of clusters is increased. The large percentage of data items that are displaced from their original clusters makes randomized DO unsuitable for clustering-based data mining applications. Randomization techniques provide good clustering for small values of $\sigma$. However, the privacy risk resulting from small offsets, as discussed in Section 4.3 makes it unfeasible to use such small offsets for data randomization.

**Table 38:** MCE % for randomized data using K-means clustering.

| Database | $[0, 1]$ | $[0, 10]$ | $[0, 20]$ | $[1, 5]$ |
|---|---|---|---|---|
| K-means $K = 10$ | | | | |
| 3000 | 0.02 | 13.5 | 11.4 | 15.5 |
| 5000 | 0.05 | 12.3 | 14.4 | 12.37 |
| 7200 | 0.05 | 18.1 | 12.32 | 11.2 |
| K-means $K = 20$ | | | | |
| 3000 | 0.04 | 0.18 | 22.3 | 31.7 |
| 5000 | 0.07 | 0.22 | 28.2 | 33.9 |
| 7200 | 0.08 | 25.3 | 32.6 | 35.4 |
| K-means $K = 30$ | | | | |
| 3000 | 0.1 | 33.2 | 45.0 | 44.5 |
| 5000 | 0.08 | 39.6 | 46.4 | 45.6 |
| 7200 | 0.07 | 30.3 | 43.7 | 42.1 |

### 6.2.7 Cluster Preservation Performance of DO Techniques

Table 40 shows a summary of the misclassification error for the different data obfuscation techniques. Random Data Perturbation (RDP) is performed by adding a noise vector of mean $\mu = 0$ and variance $\sigma^2 = 100$. The angle of rotation for rotation-based geometric transformation is 89.4 degrees. The value of $k$ for *NeNDS* as well as *GT-NeNDS* is computed by finding the average performance for $NH$ = [50, 100, 150, 300, 1000]. The size of the database used for comparison is $N = 5,000$, and the inherent clustering factor $C_{in} = 10$. The error percentages resulting from k-means and hierarchical clustering are comparable,

**Table 39:** MCE % for randomized data using Hierarchical Clustering.

| Database | $[0,1]$ | $[0,5]$ | $[0,20]$ | $[1,5]$ |
|---|---|---|---|---|
| Hierarchical Clustering $K = 10$ | | | | |
| 3000 | 0.0 | 13.1 | 14.2 | 13.2 |
| 5000 | 0.08 | 12.1 | 13.1 | 13.8 |
| 7200 | 0.07 | 12.0 | 13.3 | 14.1 |
| Hierarchical Clustering $K = 20$ | | | | |
| 3000 | 0.02 | 19.1 | 18.4 | 20.4 |
| 5000 | 0.07 | 23.8 | 22.4 | 21.5 |
| 7200 | 0.06 | 21.9 | 24.8 | 25.4 |
| Hierarchical Clustering $K = 30$ | | | | |
| 3000 | 0.03 | 32.3 | 45.0 | 41.8 |
| 5000 | 0.05 | 36.4 | 41.7 | 45.5 |
| 7200 | 0.05 | 33.6 | 45.2 | 42.7 |

and an average of the two results is used in the table. The table provides a comparison of the misclassification error as a percentage. It is observed that *RDP* performs poorly for all cluster sizes, whereas the other obfuscation techniques are comparable. Although *rotation* provides the smallest error percentage, its vulnerability to reverse engineering makes it unusable for the data obfuscation of sensitive data. The performance of the hybrid data obfuscation approach is observed to be almost as good as geometric transformations. The robust privacy-preservation capability of GT-NeNDS makes it a more suitable candidate for data protection. The performance of the obfuscation techniques degrade if the number of clusters required is chosen as a number much larger than the inherent clustering of the data, as can be noted in the case where the number of clusters is 20. This is twice the value of $C$. The loss of information in this case is a necessary condition for privacy preservation to prevent individual records from being exposed. The results of the preliminary analysis indicate that *NeNDS* and $GT - NeNDS$ provide cluster-preserving obfuscated data that is difficult to reverse-engineer.

The experimental analysis provided here shows that the cluster-preservation capability of *NeNDS* is comparable to the inherent cluster-preserving geometrical transformations.

**Table 40:** Comparison of misclassification error %.

| Obfuscation - Clusters | RDP [0,10] | RDP [0,1] | Rotation Random | NeNDS Average | GT-NeNDS Average |
|---|---|---|---|---|---|
| 2 | 3.1 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 8.3 | 0.02 | 0.02 | 0.01 | 0.02 |
| 5 | 13.5 | 0.03 | 0.04 | 0.04 | 0.04 |
| 10 | 18.1 | 0.05 | 0.13 | 0.14 | 0.14 |
| 20 | 22.3 | 0.18 | 0.45 | 0.51 | 0.59 |
| 40 | 42.1 | 0.25 | 0.87 | 0.92 | 0.95 |
| 60 | 47.3 | 0.21 | 1.12 | 1.64 | 1.71 |

The usability of a data obfuscation technique is defined in terms of its preservation of statistical distribution characteristics as well as cluster-preservation capability. An ideal obfuscation technique would be one that preserves multi-variate distribution characteristics, but such a technique would be vulnerable to privacy breaches. The next important statistical characteristics to be preserved are marginal distributions. *NeNDS* preserves marginal distributions of variables because the data is not "modified'.' The cluster-preservation capability of *NeNDS* is analyzed experimentally in this section and proved to be as good as geometrical transformations. The robustness of the privacy preservation of *NeNDS* is studied in Section 6.1.3. Although *NeNDS* falls under the non-linear bijective transformation, the large fraction of *minimum information* required for complete as well as partial reversal strengthens the privacy-preservation capability of this technique, making it very difficult to reverse engineer.

## *6.3 Comparison of Data Obfuscation Techniques*

The DO metrics *Data Usability* and *Data Privacy* proposed in this section measure the strength of a DO technique. Table 41 gives a detailed comparison of the performance of different DO techniques. The DO techniques are listed in the first column of the table.

The techniques that are compared here are Data randomization with small and large offsets (Random-Low, Random-High), Data Anonymization with small and large values for $k$ (k-Anon-Low, k-Anon-High), Data Swapping, NeNDS, Geometric transformations (Geo-Trans), and GT-NeNDS. The metrics used for comparison form the rows of the table. The *Displacement* metric indicates the similarity between the absolute values of the original and obfuscated data. A low displacement implies high vulnerability to approximate privacy breaches. The *Reversibility* metric evaluates the amount of information required for retrieving the original data from the obfuscated data. The *Stat* metric evaluates the extent to which the statistical distributions of the original data are maintained. The cluster preservation property of the DO techniques is measured by the *Cluster* metric.

**Table 41:** Comparison of DO techniques.

| **Obfuscation** | **Displacement** | **Reversibility** | **Stat** | **Cluster** |
|---|---|---|---|---|
| **Random-Low** | Very Low | Very Difficult | Good | Fair |
| **Random-High** | High | Very Difficult | Good | Poor |
| **k-Anon-Low** | Very Low | Easy | Good | Fair |
| **k-Anon-High** | High | Difficult | Fair | Poor |
| **Data Swapping** | Low | Moderate $\lfloor \frac{N}{2} \rfloor$ | Moments | Good |
| **NeNDS** | Low | Difficult $\lfloor \frac{cN}{c+1} \rfloor$ | Moments | Very Good |
| **Geo-Trans** | High | Easy $d+1$ | Poor | Very Good |
| **GT-NeNDS** | High | Difficult $> \lfloor \frac{cN}{c+1} \rfloor$ | Poor | Very Good |

Data randomization with small offsets (Random-Low) and high offsets (Random-High) is robust to absolute reversibility. The small offset of Random-Low makes it vulnerable to approximate privacy invasion and unsuitable for applications where approximate information is considered a breach. The large offset of Random-High makes it unsuitable for data

mining applications because of the distortion of the original clusters. Data anonymization for small values of $k$(k-Anon-Low) and large values of $k$(k-Anon-High) perform similar to Data randomization and are unsuitable for data that are used for data mining applications. The DO techniques Data swapping, NeNDS, Geometric transformations, and GT-NeNDS can be used for obfuscation in data mining applications. Data swapping and *NeNDS* are vulnerable to approximate reversal. Data swapping is vulnerable to absolute reversibility only if $\lfloor \frac{N}{2} \rfloor$ of the data elements in a database of size $N$ are known *a priori*. The amount of *a priori* information that leads to complete reversal of a neighborhood of *NeNDS* is $\lfloor \frac{cN}{c+1} \rfloor$, where $N$ is the number of records in the database, and $c$ is the size of each neighborhood that is permuted. Reversal of an entire dataset requires the knowledge of the permutation pattern of all the neighborhoods into which the data is distributed. Geometric transformations offer very little resistance to privacy and are unsuitable for use by themselves. GT-NeNDS, which combines *NeNDS* and Geometric transformations, provides robust protection against approximate privacy invasion as well as absolute reversibility. GT-NeNDS preserves the original clusters and also preserves moments over individual datasets.

The comparison chart for the different DO techniques indicates that a single DO technique cannot be used as a universal solution for all databases. The proposed metrics provide a baseline for selecting the optimum DO technique for a given database application.

# CHAPTER VII

# PRIVACY PRESERVING COLLABORATIVE FILTERING

Automated Collaborative Filtering (CF) applications are being widely used by E-commerce applications for predicting items that might be of interest to users based on the recommendations of other users with similar purchase histories. CF systems store inventories of items, transaction records of items purchased by users, ratings provided by users for items purchased by them, as well as demographic information of the users themselves. Some of the information fields in the User database include *Age, Location, Occupation*, which may be used to identify individual users and thus violating personal privacy. Furthermore, the lack of sufficient safeguards over the protection of user data raises concerns over individual privacy and prevents users from providing more than the minimal amount of information required to access the E-commerce services.

## *7.1 The Privacy Framework*

The model for privacy preserving collaborative filtering is explained in detail in this section. The privacy framework serves as a wrapper that obfuscates the relevant fields of data before they are fed to the CF engine. A diagrammatic view of the model is shown in Figure 17 using an example having three meta-store fronts $[MS1, MS2, MS3]$ such as *Amazon*, *C-net*, *Yahoo* that wish to share information in a privacy preserving way. Each $MS_i$ has three databases, a *User-info* database that stores demographic information regarding its users, an *Item-info* database that stores information regarding the items in its inventory, and a *Ratings-info* database that stores information regarding the ratings provided by the users on the items purchased. The databases are obfuscated and sent to the central CF server. The CF engine combines the information from all three meta-store fronts and creates three aggregated databases as shown. Recommendations are made for all the unrated items for each record in the ratings database. The aggregate database is then divided back into the

three individual databases, which are now populated with recommendations for unrated items. The databases are then sent back to the meta-store fronts. The stores provide recommendations to their users based on the results obtained from the CF engine. Since the databases are dynamic in nature, the $MS_i$ obfuscate the updated databases periodically and send them to the CF server so that the recommendations are made on the most recent ratings made by individuals.

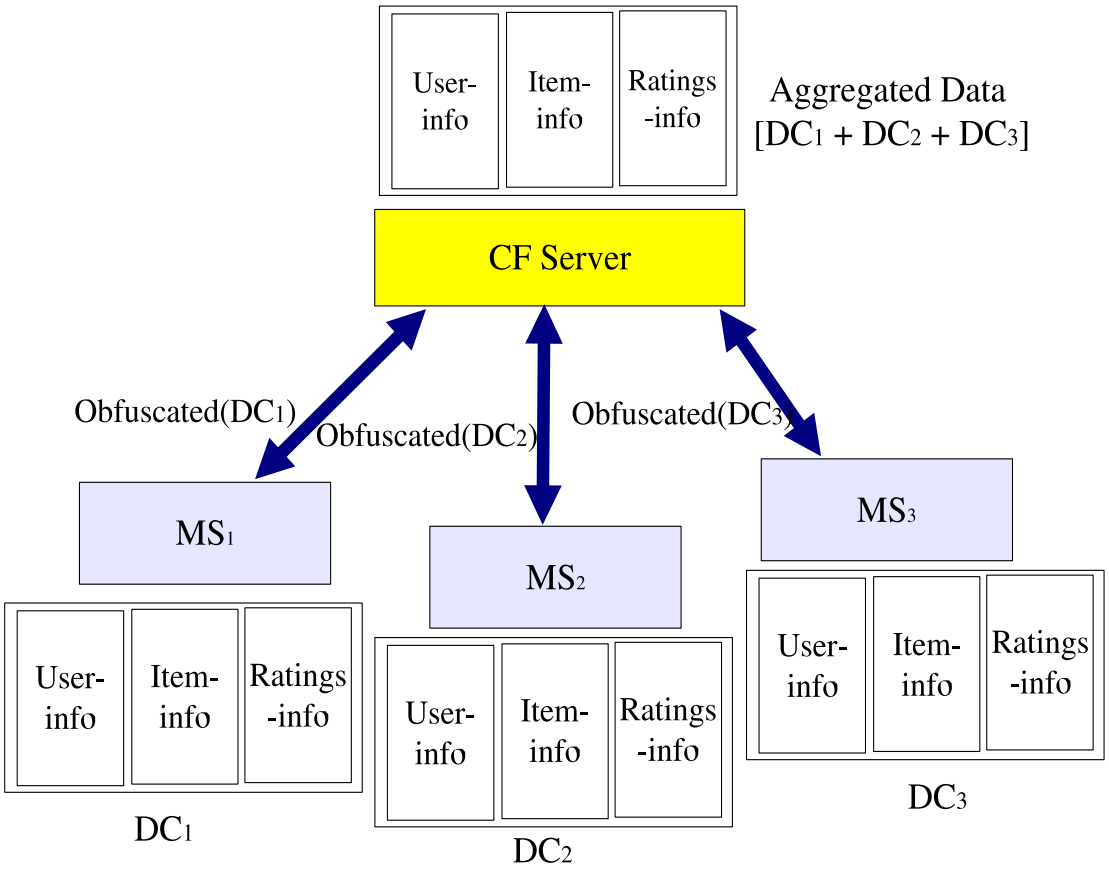



**Figure 17:** Privacy preserving framework for CF.

### 7.1.1 Data Selection

Figure 17 shows that the CF engine performs predictions on the obfuscated data. For the recommendations to be accurate, the obfuscated data should preserve maximum information content while still ensuring data privacy. In this section, the problem of privacy is analyzed with respect to each database to determine which of the fields need to be obfuscated. Each

field in the database is referred to as a dataset. The decision to obfuscate a dataset is made based on the following questions

1. *Can the dataset be used to single out an individual in the database?* An example of such a dataset is the social-security number or the name of an individual. Such information is highly susceptible to privacy breach and should be removed from the database if possible, or at least obfuscated with a robust data obfuscation technique.

2. *Does the dataset contain any information that can be used in combination with other datasets to identify an individual?* Fields such as *Age* and *Location* in any database can be used in conjunction with other fields to identify one or more records. Such fields can lead to indirect privacy breach and need to be obfuscated.

3. *Can a priori knowledge of some of the entries in the dataset lead to identification of an individual's entire record?*. Knowledge of an individual's exact preference on a set of items can be used to track down personal information of the individual. Such fields need to be obfuscated to protect the identity of the individuals in the database.

If the answer to any of these questions is *Yes*, then it needs to be obfuscated. The CF engine uses three databases for prediction, *User-info, Ratings-info, Item-info.* The first database, *User-info* contains demographic information of each user, which includes the [**User-id, Name, Age, Location, ...**]. All of these fields pose a privacy threat and need to be protected. Some of these fields, such as the **[Age, Location]** are likely to be used by the CF engine for optimized prediction and need to be obfuscated. Fields such as the **[Name]** are never used and should be removed from the database. The second database, *Ratings-info* stores the ratings provided by individual users. The ratings do not hold any personal information by themselves. However, if a person's exact ratings for a set of items are known, the corresponding user-id from the ratings database can be used to retrieve other information regarding the individual and to target the individual with subsequent attacks. All the ratings need to be obfuscated.

## *7.2 NeNDS-based Collaborative Filtering*

Several approaches have been proposed for privacy preserving data mining applications. Random data perturbation [5] and data anonymization [62] are some commonly used data obfuscation techniques for applications where aggregate statistics are sufficient. These approaches protect data by adding random noise to them or by a process of suppression and generalization. The lossy nature of the transformations destroys the inherent clustering in the data, making them unsuitable for applications that use classification or cluster-based data mining. Geometric transformations [44][45] and data swapping [51] preserve clustering, but offer weak privacy preservation of the data, which renders them unsuitable for sensitive applications [46]. In NeNDS, each field of the database is treated separately, and the datasets are obfuscated by permuting sets of similar items. The permutation process ensures lossless transformation and also offers a stronger transformation than data swapping. Permutation among similar elements ensures that the clusters are preserved. A comparison of the strength of the data obfuscation techniques with respect to privacy protection and data usability is presented in [46]. The results show that NeNDS offers robust data privacy as well as data usability. The approach can be applied on any dataset that forms a metric space. One drawback of NeNDS is that the transformed data might be close enough to the original value to be considered vulnerable. This vulnerability is fixed by performing a geometric transformation such as rotation, scaling, or translation on the NeNDS-obfuscated data. The linearity property of geometric transformations preserves clustering and changes the values of the individual data. The weakness of geometric transformations is taken care of by performing NeNDS-based data obfuscation as a first step. This hybrid-NeNDS approach is used here to obfuscate the data for CF.

The following example demonstrates the process of data obfuscation using hybrid-NeNDS. Table 42 represents a *User-info* database containing records of 8 users with 3 fields [**User-ID, Name, Age**]. Among these fields, the User-ID field is retained without obfuscation.

The [**Name field**] can be removed as it not necessary for CF. The [**Age**] field is a necessary field and is subjected to NeNDS as follows. The first step in NeNDS is the creation

**Table 42:** User-info database.

| User-ID | Name | Age |
|---|---|---|
| 1 | John Smith | 42 |
| 2 | Alan Finn | 65 |
| 3 | Ann Taylor | 40 |
| 4 | Corey Meyers | 35 |
| 5 | Fred Sutton | 68 |
| 6 | Jack Thomas | 75 |
| 7 | Jenny Higgins | 37 |
| 8 | Rene Robinson | 52 |

of neighborhoods. The data are divided into two neighborhoods of similar elements, each consisting of 4 elements. The first neighborhood $NH_1$ consists of the ages $[35, 37, 40, 42]$ and neighborhood $NH_2$ consists of ages $[52, 65, 68, 75]$. The data items in each neighborhood are permuted based on the nearest neighbor algorithm 5.2.

**Table 43:** Transformed User-info database.

| **User-ID** | *Age′* | *Age″* |
|---|---|---|
| 1 | 40 | 28 |
| 2 | 52 | 36 |
| 3 | 35 | 25 |
| 4 | 37 | 26 |
| 5 | 75 | 53 |
| 6 | 65 | 46 |
| 7 | 42 | 29 |
| 8 | 68 | 48 |

Table 44 represents a *Ratings-info* database with ratings of the 8 users on 2 items. The ratings are on a scale of $[1-10]$. These ratings are first transformed using NeNDS by creating 2 neighborhoods for each dataset and then permuting the data in each neighborhood. The NeNDS transformed data are presented in Table 45. The missing entries in the database are not included in any neighborhood and are retained even after transformation. The NeNDS-transformed database is then scaled by a factor of 0.8 on both fields. The transformed database is shown in Table 46.

**Table 44:** Ratings-info database.

| Item-ID | Rating-1 | Rating-2 |
|---|---|---|
| 1 | 4 | 3.5 |
| 2 | 5.5 | 4.1 |
| 3 | | 2.5 |
| 4 | 9 | 7.5 |
| 5 | 8.5 | 8 |
| 6 | 4.5 | |
| 7 | 9.5 | 9 |
| 8 | 10 | 9.5 |

**Table 45:** NeNDS transformed Ratings-info database.

| Item-ID | Rating-1 | Rating-2 |
|---|---|---|
| 1 | 4.5 | 2.5 |
| 2 | 4 | 3.5 |
| 3 | | 4.1 |
| 4 | 8.5 | 9 |
| 5 | 10 | 7.5 |
| 6 | 5.5 | |
| 7 | 9.5 | 9.5 |
| 8 | 9 | 8 |

**Table 46:** Scaled-NeNDS transformed Ratings-info database.

| Item-ID | Rating-1 | Rating-2 |
|---|---|---|
| 1 | 3.6 | 2 |
| 2 | 3.2 | 2.8 |
| 3 | | 3.2 |
| 4 | 6.8 | 7.2 |
| 5 | 6.4 | 6 |
| 6 | 3.8 | |
| 7 | 7.6 | 7.6 |
| 8 | 7.1 | 6.4 |

The *Item-info* database is retained in the unobfuscated form. All three databases *Obf-User-info*, *Obf-Rating-info*, and *Item-info* are sent to the CF server for predicting recommendations for items that have not yet been rated by the users. The result of the CF engine

is shown in Table 47. These ratings are then used for making recommendations to the user. The ratings can be stored and used in its scaled form or scaled back to its original range. Table 47 contains the predictions on the unrated items for the unobfuscated database. The predictions for the unrated items on the obfuscated database $P_{obf} = [3.8, 3.3]$ correspond to $P'_{obf} = [5.4, 4.7]$ when scaled back by a factor of 0.7. These ratings are similar to the predicted values on the unobfuscated data $P_{orig} = [4.5, 4]$. Since this is just an example with very few data items, the results in the two cases do not match exactly. The difference is almost negligible for larger datasets.

**Table 47:** Prediction results for obfuscated data.

| **Item-ID** | $Rating - 1$ | $Rating - 2$ |
|---|---|---|
| 1 | 3.6 | 2 |
| 2 | 3.2 | 2.8 |
| 3 | 3.8 | 3.2 |
| 4 | 6.8 | 7.2 |
| 5 | 6.4 | 6 |
| 6 | 3.8 | 3.3 |
| 7 | 7.6 | 7.6 |
| 8 | 7.1 | 6.4 |

**Table 48:** Prediction results for unobfuscated data.

| **Item-ID** | $Rating - 1$ | $Rating - 2$ |
|---|---|---|
| 1 | 4 | 3.5 |
| 2 | 5.5 | 4.1 |
| 3 | 4.5 | 2.5 |
| 4 | 9 | 7.5 |
| 5 | 8.5 | 8 |
| 6 | 4.5 | 4 |
| 7 | 9.5 | 9 |
| 8 | 10 | 9.5 |

The accuracy of the predictions for large databases is studied in Section 8.2. For shared CF, each meta-store front performs a NeNDS transformation on the data followed by a geometric transformation using the same parameters. The parameters for the geometric

transformations can be decided by the central server, or by a secure token exchange among the meta-store fronts. Rotation-based transformations cannot be used here because of the presence of incomplete records. Rotation of a record with a missing entry results in a transformed record that has a non-zero value in place of the missing entry. Rotation of such records distorts the relative distances between records. CF systems using similarity measures for predicting user preference would fail if the relative distances are altered significantly. Since most of the databases used for CF are sparse databases, rotation-based transformations are not feasible. The scaling transformation discussed here can be replaced by any linear transformation vector that is not affected by missing entries.

# CHAPTER VIII

# PERFORMANCE ANALYSIS: NENDS-BASED CF

## *8.1 Collaborative Filtering Algorithms*

Several collaborative filtering approaches have been developed for recommendation systems. Automated CF systems are widely used for providing recommendations to users based on the ratings of users with similar interests. The different CF systems can be broadly classified into memory-based CF, model-based CF, and hybrid memory-model CF. Memory based systems use the raw data in the database by applying nearest neighbor techniques for predicting user preferences. Model-based approaches first create a model based on the available information and use this model to make probabilistic predictions for the unrated items. Hybrid approaches use the model based approach to create sets of similar users. The predictions are then made by using memory-based techniques on the set of similar users, thus optimizing the accuracy and time complexity of the predictions. In this paper, we use the Pearson's correlation co-efficient and Vector similarity algorithms, which are memory based approaches. We also use the personality diagnosis algorithm to analyze its performance on obfuscated data.

### 8.1.1 Pearson Correlation

This was the first automated CF technique introduced by GroupLens [52] to provide personalized recommendations for Usenet news articles. This is a memory-based approach where the Pearson correlation is used to weight user similarity. All the correlated neighbors are used in the prediction. The prediction for an active user is based on the weighted average of the deviation from the mean of the neighbors. In Equation 26 $p_{a,i}$ is the final prediction for an active user $i$ on item $a$, $n$ represents the number of neighbors, $j$ represents an individual neighbor, and $w_{a,j}$ is the similarity weight between the user and the neighbor computed for item $a$. The weight is determined from Pearson's correlation co-efficient

shown in Equation 27.

$$p_{a,j} = \overline{r_a} + \frac{\sum_{i=1}^{n}[(r_{u,i} - \overline{r_u}) * w_{a,u}]}{\sum_{i=1}^{n} w_{a,u}} \tag{26}$$

$$w_{a,u} = \frac{\sum_{i=1}^{m}[(r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})]}{\sqrt{\sum_{i=1} m(r_{a,i} - \overline{r_a})^2 \sum_{i=1}^{m} (r_{u,i} - \overline{r_u})^2}} \tag{27}$$

### 8.1.2 Vector Similarity

The vector similarity algorithm was proposed for information retrieval. It was first used by GroupLens for collaborative filtering. Here each user's ratings are treated as a vector, and similarity weights are computed based on the cosine of the angle between the ratings vectors. The weights are computed based on the Equation 28, where $r_{aj}$ is the rating for item $j$ by active user $a$, and $I_a$ is the set of items for which ratings have been provided by the active user a. The terms in the denominator are squared to normalize the votes so that users are given the same weight regardless of the number of titles that they provide ratings for.

$$w_{a,i} = \sum_{j} \frac{r_{aj}}{\sqrt{[\sum_{k \in I_a} {r_{ak}}^2]}} \frac{r_{ij}}{\sqrt{[\sum_{k \in I_i} {r_{i,k}}^2]}} \tag{28}$$

### 8.1.3 Personality Diagnosis

Personality diagnosis based CF was proposed by [49]. This approach takes advantage of the benefits of memory and model based approaches. Model-based approaches operate by creating a probabilistic model of the data, following which predictions are made using just the model. The advantage of this method is the reduced memory usage with respect to the memory-based approaches. However, developing the model is time-intensive. Each time a data point needs to be added to the ratings, the model needs to be recalculated. The personality diagnosis approach uses a probabilistic model to compute the personality type of a user based on the true ratings of the user. The second iteration involves comparing the preferences of the active user with all other users who have a similar personality type using a memory-based approach.

## 8.2 Experimental Evaluation

The performance of the privacy framework using hybrid-NeNDS is discussed in this section. The experiments compare the prediction results of the obfuscated data with the prediction results of the original data. Two sets of tests are performed to study the performance of the privacy framework. The first set involves the accuracy of predicting an item one at a time to a user. The second test studies the results when an ordered set of recommendations are provided to a user. The amount of distortion in the results due to data obfuscation is analyzed here.

The first step of the experiment involves dividing the data (users and their ratings) into a training set and a test set. The training set is used as the database for the CF engine. Each user/ratings record in the test set is iteratively presented to the CF engine for making predictions. The ratings of the test user, known as the *active* user are divided into a set of observed ratings, $I_a$ and a set of unrated ratings $P_r$. The ratings $I_a$ are presented to the CF engine and the predicted ratings $P_{CF}$ for the unrated items are compared with the set $P_r$.

The first set of tests to compare the performance of one-at-a-time recommendations are measured by using the average absolute deviation of the predicted ratings $p_i$ with respect to the actual ratings on items for which the test set users have entered ratings ($r_i$). This metric was first introduced in GroupLens [52] and is used as a standard for comparing CF systems. The mean absolute deviation for a single user on $m_a$ predicted items is given by Equation 29. The error is averaged over all the users in the test set. Since the two data collections used here have different ranges for ratings, the normalized mean absolute error $NMAE$ is evaluated [22] as shown in Equation 30.

$$|\overline{E}| = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{29}$$

$$|NMAE| = \frac{|\overline{E}|}{r_{max} - r_{min}} \tag{30}$$

The second set of tests, which measure the ranking sequence of the ordered list of recommendations, is evaluated using the rank scoring metric proposed in [33]. The metric

extends the binary precision-recall metric used in information retrieval to estimate the likelihood that the user selects an item on the ordered list. The metric represents the expected utility of a ranked list of items by Equation 31. The list of items are sorted by index $j$ in the decreasing order of the rating $r_{a,j}$. The parameters $d$ denote the neutral vote and is applied to items for which user rating is not available. The term $\alpha$ represents the number of the item on the list such that the likelihood of the user viewing it is 50%.

$$R_a = \sum_j \frac{max(r_{\alpha,j} - d, 0)}{2^{(j-1)/(\alpha-1)}} \tag{31}$$

The final score for the set of active users is given in equation 32, where ${R_a}^{max}$ is the maximum achievable utility which is achieved when all the observed items appeared at the top of the ordered list of recommendations. This equation normalizes the results and makes it independent of the test set and number of items.

$$R = 100 \frac{\sum_a R_a}{\sum_a {R_a}^{max}} \tag{32}$$

### 8.2.1 Data Sets

The evaluation considers two different database collections.

1. BookCrossing [70]: This collection consists of three databases [**User-info, Book-info, and Ratings-info**]. The **User-info** database contains demographic information of 278,858 users [**ID, Location, Age**]. The [**Book-info**] database has information regarding the title, ISBN, year of publication, author, publisher, and edition for 271,379 books. The [**Ratings-info**] database contains a total of 1,149,780 ratings by the listed users for the books specified in the database. The fields that are obfuscated are: [**User-info: Age**] and all the fields in the [**Ratings-info**] database.

2. Movielens [25]: The three databases [**User-info, Movie-info, and Ratings-info**] form this database. The **User-info** database contains demographic information of 6040 users [**ID, Age, gender, occupation, zip**]. The [**Movie-info**] database has information regarding the Movie-ID, title, release date, and video release date for 3,900 movies. The [**Ratings-info**] database contains a total of 2,811,983 ratings by

the listed users for the movies specified in the database. The fields that are obfuscated are: [**User-info: Age, zip code**]. The gender and occupation fields are removed from the database before sending it to the CF server. All the fields in the [**Ratings-info**] database are obfuscated.

### 8.2.2 Experiment Results

To evaluate the performance of the CF engine, we carried out three types of tests for the one-at-a-time and ordered-list recommendations. The All-but-one test provides all the ratings except one for each active user in the test set. The accuracy of prediction of the single rating is measured in this test. In the Given-2 test, the observed ratings set $I_a$ contains only two ratings. The accuracy of predictions of the rest of the ratings in the unrated set $P_r$ is analyzed here. Given-10 measures the accuracy of the predictions with 10 ratings in the active user's observed-ratings set.

The data collections are arbitrarily divided into three sets, each set representing the repository of one meta-store front. All three repositories are first obfuscated using NeNDS, where each data set was divided into 100 neighborhoods. All three repositories apply the same geometric transformation to the data. For the User-info data, a scaling transformation of 0.8 was applied for each field. The ratings-info database was transformed with a different scaling vector that was generated randomly. The resulting databases were then appended to form a single collection. The collection was then divided into a training set and test set in the ratio 75% : 25%. The training/test sets for the two data collections are shown in table 49. Each of the entries in the test set is then added to the training set one at a time to determine the mean absolute error and ranking score for all three tests. The tests are performed three times, once with each CF algorithm: Pearson, Vector Similarity, and Personality diagnosis.

**Table 49:** Training and test set partitions.

| Data collection | Training Set | Test Set |
|---|---|---|
| **Bookcrossing** | 209144 | 69714 |
| **Movielens** | 4530 | 1510 |

Table 50 shows the prediction results of the three algorithms for the 'All-but-1' case.

The results obtained for the individual algorithms with original data match the results obtained in [33]. The normalized mean absolute error for the obfuscate data are consistent with the results for the original data. This shows that the privacy framework does not affect the CF for the all-but-1 case.

**Table 50:** Prediction accuracy for the All-but-one test.

| CF Algorithm | Original Data | Obfuscated Data | Error % |
|---|---|---|---|
| **Pearsons Movielens** | 0.198 | 0.198 | 0.0 |
| **V. Similarity Movielens** | 0.241 | 0.242 | 0.1 |
| **P. Diagnosis Movielens** | 0.192 | 0.193 | 0.1 |
| **Pearsons Bookcrossing** | 0.201 | 0.202 | 0.1 |
| **V. Similarity Bookcrossing** | 0.211 | 0.211 | 0.1 |
| **P. Diagnosis Bookcrossing** | 0.201 | 0.203 | 0.2 |

The results for the Given-2 test are shown in Table 51. It is observed that the predicted results on the obfuscated data have small differences from the predictions using original data in a few cases. The difference for the vector similarity algorithm is slightly higher than for the other two algorithms. However, the error introduced by the data obfuscation is around 2% in the worst case, which should be acceptable given the additional benefit from privacy preservation.

Table 52 contains the results for the Given-10 test. The results in this test indicate that the errors introduced in this case are much smaller than the errors introduced when only two ratings were provided to the CF engine. Two of the three algorithms yield similar results with and without data obfuscation. The vector similarity algorithm produces results that are different from the original predictions, but the performance is better than the Given-2 case. The performance of the algorithms with increasing number of Given-n ratings was evaluated. The error difference between the original and obfuscated results decrease exponentially with the increase in number of Given ratings.

**Table 51:** Prediction accuracy for the Given-2 test.

| CF Algorithm | Original Data | Obfuscated Data | Error % |
|---|---|---|---|
| **Pearsons Movielens** | 0.228 | 0.229 | 0.1 |
| **V. Similarity Movielens** | 0.291 | 0.312 | 2.1 |
| **P. Diagnosis Movielens** | 0.209 | 0.211 | 0.2 |
| **Pearsons Bookcrossing** | 0.231 | 0.232 | 0.1 |
| **V. Similarity Bookcrossing** | 0.247 | 0.262 | 1.5 |
| **P. Diagnosis Bookcrossing** | 0.213 | 0.215 | 0.2 |

**Table 52:** Prediction accuracy for the Given-10 test.

| CF Algorithm | Original Data | Obfuscated Data | Error % |
|---|---|---|---|
| **Pearsons Movielens** | 0.199 | 0.200 | 0.1 |
| **V. Similarity Movielens** | 0.208 | 0.209 | 0.1 |
| **P. Diagnosis Movielens** | 0.196 | 0.196 | 0.0 |
| **Pearsons Bookcrossing** | 0.201 | 0.202 | 0.1 |
| **V. Similarity Bookcrossing** | 0.237 | 0.239 | 0.2 |
| **P. Diagnosis Bookcrossing** | 0.197 | 0.201 | 0.4 |

The results for the ranking score for ordered recommendations are tabulated in Tables 53,54 for the Given-2 and Given-10 tests respectively. The ranking score is a percentage that measures the utility of the items recommended based on the order of recommendation. A higher value indicates better performance. It is observed that a ranking score of 48.4 is achieved using Pearsons correlation in the Movielens data and a score of 36.9 is achieved in the case of the Bookcrossing data when 10 ratings are provided to the CF for each active user. The performance of the CF recommendations is better when more ratings are available for prediction. The performances of the CF algorithms on obfuscated data are observed

to follow the original predictions closely. Data obfuscation affects the results of the rank scoring by 5% on the average, and performs the worst for the hybrid model. The reason for the distortion can be attributed to the fact that the actual ratings are distorted, which causes the highest rating to be substituted by a lower value. As a result, the corresponding recommendation is moved to a lower rank in the recommended list.

**Table 53:** Performance based on the rank scoring test.

| CF Algorithm | Original Data | Obfuscated Data | Error % |
|---|---|---|---|
| **Pearsons Movielens** | 42.8 | 40.5 | 2.3 |
| **V. Similarity Movielens** | 36.5 | 35.2 | 1.3 |
| **P. Diagnosis Movielens** | 39.6 | 31.8 | 7.8 |
| **Pearsons Bookcrossing** | 33.9 | 31.7 | 2.2 |
| **V. Similarity Bookcrossing** | 31.4 | 30.5 | 0.9 |
| **P. Diagnosis Bookcrossing** | 29.2 | 25.1 | 4.1 |

**Table 54:** Performance based on the rank scoring test.

| CF Algorithm | Original Data | Obfuscated Data | Error % |
|---|---|---|---|
| **Pearsons Movielens** | 48.4 | 44.5 | 3.9 |
| **V. Similarity Movielens** | 39.5 | 35.2 | 4.3 |
| **P. Diagnosis Movielens** | 45.6 | 40.8 | 4.8 |
| **Pearsons Bookcrossing** | 36.9 | 34.7 | 2.2 |
| **V. Similarity Bookcrossing** | 32.4 | 30.5 | 1.9 |
| **P. Diagnosis Bookcrossing** | 38.2 | 31.1 | 7.1 |

Tests were also conducted by applying random data perturbations to the original data before applying the CF algorithms. The tests are carried out for two sets of random distribution parameters: $[\mu_1, \sigma_1] = [0, 0.5]$ and $[\mu_2, \sigma_2] = [0, 10]$, where $[\mu, \sigma]$ represent the

mean and standard deviation for the normal distribution. The results for the All-but-one, Given-2, and Given-10 tests are provided in Tables 55, 56, and 57 respectively. In each of the three tests, the performances of the CF algorithms are comparable to the original predictions for the data randomized using $[\mu_1, \sigma_1] = [0, 0.5]$.

**Table 55:** Prediction accuracy for the All-but-one test.

| **CF Algorithm** | **Rand. Data** $[\mu_1, \sigma_1]$ | **Rand. Data** $[\mu_2, \sigma_2]$ | **Error** % | **Error** % |
|---|---|---|---|---|
| **Pearsons Movielens** | 0.198 | 0.598 | 0.0 | 40.0 |
| **V. Similarity Movielens** | 0.245 | 0.621 | 0.4 | 38.0 |
| **P. Diagnosis Movielens** | 0.192 | 0.432 | 0.0 | 24.0 |
| **Pearsons Bookcrossing** | 0.204 | 0.501 | 0.3 | 49.7 |
| **V. Similarity Bookcrossing** | 0.211 | 0.634 | 0.0 | 41.7 |
| **P. Diagnosis Bookcrossing** | 0.201 | 0.453 | 0.0 | 25.2 |

**Table 56:** Prediction accuracy for the Given-2 test.

| **CF Algorithm** | **Rand. Data** $[\mu_1, \sigma_1]$ | **Rand. Data** $[\mu_2, \sigma_2]$ | **Error** % | **Error** % |
|---|---|---|---|---|
| **Pearsons Movielens** | 0.228 | 0.399 | 0.0 | 17.1 |
| **V. Similarity Movielens** | 0.292 | 0.493 | 0.1 | 20.2 |
| **P. Diagnosis Movielens** | 0.211 | 0.506 | 0.2 | 29.7 |
| **Pearsons Bookcrossing** | 0.235 | 0.552 | 0.4 | 32.1 |
| **V. Similarity Bookcrossing** | 0.247 | 0.426 | 0.0 | 17.9 |
| **P. Diagnosis Bookcrossing** | 0.213 | 0.548 | 0.0 | 33.5 |

The results for second random distribution $[\mu_2, \sigma_2] = [0, 10]$ are significantly worse than the original predictions in all the three tests. The reason for this behavior is that the noise distribution distorts the relative distances among the individual data items. The accuracy

**Table 57:** Prediction accuracy for the Given-10 test.

| **CF Algorithm** | **Rand. Data** $[\mu_1, \sigma_1]$ | **Rand. Data** $[\mu_2, \sigma_2]$ | **Error** % | **Error** % |
|---|---|---|---|---|
| **Pearsons Movielens** | 0.199 | 0.30 | 0.0 | 10.1 |
| **V. Similarity Movielens** | 0.21 | 0.298 | 0.2 | 9.0 |
| **P. Diagnosis Movielens** | 0.197 | 0.496 | 0.1 | 30.0 |
| **Pearsons Bookcrossing** | 0.201 | 0.302 | 0.0 | 10.1 |
| **V. Similarity Bookcrossing** | 0.240 | 0.443 | 0.3 | 20.6 |
| **P. Diagnosis Bookcrossing** | 0.201 | 0.321 | 0.4 | 12.4 |

of prediction for CF algorithms on randomly perturbed data is dependent on the type of distribution and range of distribution for the random variable. The random distribution range cannot exceed a threshold value. However, this restriction contradicts the requirement of data privacy. If the noise distribution is not sufficiently large to do a good job of hiding the original values, the datasets become vulnerable to approximate privacy breach [46]. The selection of a distribution range for random perturbation is a critical factor that affects the privacy and usability of data. The only input parameter for NeNDS is the neighborhood size $NH$. In [46], the sensitivity of the neighborhood size $NH$ on the *Misclassification error* (MCE) in clustering was evaluated. The results show that the performance of NeNDS is insensitive to the parameter $NH$. We expect NeNDS to perform similarly for the different test scenarios and are currently evaluating the effect of varying $NH$ sizes on the accuracy of the predicted results. The ability of hybrid-NeNDS to provide privacy without trading off usability of the CF system makes it an excellent candidate for privacy protection of data used for CF.

## *8.3 Effect of Neighborhood Size on NeNDS-based Filtering*

All the experiments to evaluate the performance of CF systems on NeNDS transformed were carried out for $NH = 100$ neighborhoods. In Chapter 6.2, it was shown that varying the

number of neighborhoods for NeNDS had little or no effect on the clustering performance of K-means and hierarchical clustering. In this section the effect of neighborhood size on the performance of collaborative filtering is evaluated. Each of the databases, MovieLens and Bookcrossing are obfuscated to produce three different databases [$DB_{50}$, $DB_{500}$, $DB_{1000}$] for $NH = [50, 500, 1000]$ neighborhoods. The obfuscated databases are evaluated for the [**All-but-1, Given-2, Given-10**]. The results of the three tests are shown in Tables 58, 59 and 60. Table 58 shows that the results of the All-but-1 test are identical in for all three obfuscated databases. The number of neighborhoods for NeNDS has no effect on the results of the All-but-1 test. In Table 59, it is observed that the results for $NH = 1000$ are slightly better than the results for $NH = [50, 500]$. However, the improvement is only $0.001-0.002\%$ in each case of the Given-2 test. The results for the Given-10 test shown in Table 60 indicate that the performance of the obfuscated data for $NH = [500, 1000]$ are marginally better than the database obfuscated with $NH = 50$. Here again, the improvement is $0.01 - 0.04$. The results for $NH = 1000$ is closer to the original database results indicating that there is a marginal improvement in the performance of the Given-10 test for larger number of neighborhoods.

**Table 58:** Effect of number of neighborhoods: All-but-one test.

| **CF Algorithm** | **Original Data** | **Obfuscated Data** | | |
|---|---|---|---|---|
| * | * | NH = 50 | NH=500 | NH=1000 |
| **Pearsons Movielens** | 0.198 | 0.198 | 0.198 | 0.198 |
| **V. Similarity Movielens** | 0.241 | 242 | 0.242 | 0.242 |
| **P. Diagnosis Movielens** | 0.192 | 0.192 | 0.191 | 0.191 |
| **Pearsons Bookcrossing** | 0.201 | 0.202 | 0.202 | 0.202 |
| **V. Similarity Bookcrossing** | 0.211 | 0.211 | 0.211 | 0.211 |
| **P. Diagnosis Bookcrossing** | 0.201 | 0.203 | 0.202 | 0.202 |

These experiments show that the number of neighborhoods chosen for NeNDS does not have a significant effect on the performance of the CF algorithms. The number of

**Table 59:** Effect of number of neighborhoods: Given-2 test.

| **CF Algorithm** | **Original Data** | **Obfuscated Data** | | |
|---|---|---|---|---|
| * | * | NH = 50 | NH=500 | NH=1000 |
| **Pearsons Movielens** | 0.228 | 0.229 | 0.229 | 0.228 |
| **V. Similarity Movielens** | 0.291 | 0.312 | 0.311 | 0.309 |
| **P. Diagnosis Movielens** | 0.209 | 0.211 | 0.211 | 0.210 |
| **Pearsons Bookcrossing** | 0.231 | 0.234 | 0.232 | 0.232 |
| **V. Similarity Bookcrossing** | 0.247 | 0.264 | 0.263 | 0.260 |
| **P. Diagnosis Bookcrossing** | 0.213 | 0.215 | 0.215 | 0.214 |

**Table 60:** Effect of number of neighborhoods: Given-10 test.

| **CF Algorithm** | **Original Data** | **Obfuscated Data** | | |
|---|---|---|---|---|
| * | * | NH = 50 | NH=500 | NH=1000 |
| **Pearsons Movielens** | 0.199 | 0.201 | 0.200 | 0.200 |
| **V. Similarity Movielens** | 0.208 | 0.210 | 0.209 | 0.208 |
| **P. Diagnosis Movielens** | 0.196 | 0.197 | 0.196 | 0.196 |
| **Pearsons Bookcrossing** | 0.201 | 0.202 | 0.201 | 0.201 |
| **V. Similarity Bookcrossing** | 0.237 | 0.239 | 0.238 | 0.237 |
| **P. Diagnosis Bookcrossing** | 0.197 | 0.201 | 0.200 | 0.200 |

neighborhoods can therefore be selected based on the size of the database and the privacy requirements for the CF system.

# CHAPTER IX

# CONCLUSION AND FUTURE WORK

This thesis proposes privacy preserving framework for collaborative filtering applications. The problem of privacy is still not a well-understood one. While there is a definite need for privacy, there is no clear-cut answer to the question of what information is considered private and when a database is considered to be breached. In general, if any information about an individual revealed from a database can be obtained in any other way without access to the database, the information is no considered as private. Gaining access to such information is not considered as a privacy breach [10]. In practical databases, this implies that any query in a database should not reveal information that could lead to the identification of an individual otherwise impossible without the database. The definition of privacy used in this dissertation is to prevent any information regarding an individual from being revealed either directly or by reverse-engineering the database. NeNDS-based transformations obfuscate individual records by permuting each dataset individually. Any query made to the database is guaranteed to reveal an answer that is close to the truth but different from the exact truth. GT-NeNDS takes privacy a step further by transforming the data to a state where the values in the database are clearly different from the original values. The inter-relationship among the data items are preserved, which makes this DO approach an excellent candidate for data mining applications.

While there has been tremendous growth in the areas of information retrieval and optimization measures for CF systems, there has been little research in the area of privacy preserving CF. Trust-based systems have been proposed to thwart targeted attacks on CF systems to promote or demote items maliciously. CF using factor analysis proposes a secure method for CF among peers. This method can only be used among a known set of users, where an active user seeks out information. This paper proposes a privacy framework that allows automated recommendations to be made to users in a privacy preserving manner

that ensures the privacy of users. The framework can be used to share information among multiple meta-store fronts for information for mutual gain. New sellers suffer an initial setback, referred to as cold-start, because of the lack of a data pool to provide recommendations to its users. The cold start problem can be averted by the presence of a shared CF engine. The experimental results indicate that the accuracy of CF engines remains nearly the same in spite of the preliminary data obfuscation process. Although the rank scoring metric indicated that the utility of the ranking order is decreased by data obfuscation, the error is only about 5% on average, which is an acceptable trade-off, given the benefits of a robust privacy-preservation mechanism.

Some interesting problems for future work are listed below.

- NeNDS-based DO can be performed only on static databases. For dynamic databases, or databases that undergo constant changes, NeNDS can be applied to the database periodically. However, this could be time-intensive for some applications because the entire database has to be obfuscated each time. An interesting problem is to study ways in which NeNDS-based obfuscation can be applied only to the parts of the database that have been modified without losing clustering information of the data.

- The DO technique proposed in this research assumes that the databases have been pre-processed and cleaned. This means that the database has no duplicate records, incomplete records, or invalid data. An un-cleaned database could distort the results of the obfuscation process. One of the pre-processing steps for data mining is data cleaning and sanitization. Data obfuscation can only be performed on sanitized databases. An interesting problem to consider is the development of a DO technique that can be used to obfuscate databases that have not been processed.

- The collaborative filtering framework proposed here assumes that the users of the E-commerce sites are valid users and are not malicious. The framework does not include mechanisms to avoid shilling or targeted attacks on the CF system. Methods such as building a web of trust, and trust-aware CF have been proposed to counter such targeted attacks. An evaluation of the performance of the NeNDS-based CF

framework that incorporates trust-based techniques is an interesting future work. Such a framework would provide a robust architecture for sharing information among multiple E-commerce vendors for centralized collaborative filtering.

# REFERENCES


[1] ACM, *"Privacy Preserving K-means Clustering over Vertically Partitioned Data"*, (Washington D.C), Aug 2003.

[2] Aggarwal, C. and Yu, P., "A Condensation Approach to Privacy Preserving Data Mining," in *Advances in Database Technology - EDBT 2004*, pp. 183–199, 2004.

[3] Agrawal, D. and Aggarwal, C., ""On the Design and Quantification of Privacy Preserving Algorithms"," in *Proc. of the conference on Special Interest Group for Management of Data*, (Santa Barbara, California), pp. 247–255, May 2001.

[4] Agrawal, R., Kiernan, J., Srikant, R., and Xu, Y., ""Order Preserving Encryption for Numeric Data"," in *Proc. of Special Interest Group on Management of Data*, (Paris, France), pp. 563–574, ACM Press, June 2004.

[5] Agrawal, R. and Ramakrishnan, S., ""Privacy-Preserving Data Mining"," in *ACM Special Interest Group on Management of Data*, pp. 439–450, 2000.

[6] http://www-2.cs.cmu.edu/awm/tutorials/kmeans.html, Feb 12, 2005.

[7] Bakken, D., Parameswaran, R., and Blough, D., ""Data Obfuscation: Anonymity and Desensitization of Usable Data Sets"," *IEEE Security and Privacy*, vol. 2, pp. 34–41, Nov-Dec 2004.

[8] Bakken, D., Parameswaran, R., and Blough, D., "Data Obfuscation: Anonymity and Desensitization of Usable data Sets," *IEEE Security and Privacy*, vol. 2, pp. 34–41, Nov-Dec 2004.

[9] Bertino, E., Jajodia, S., and Samarati, P., "Supporting multiple access control policies," in *IEEE Symposium on Security and Privacy*, (Oakland, CA), pp. 94–109, IEEE Society Computer Press, 1996.

[10] Blum, A., Dwork, C., McSherry, A. F., and Nissim, K., "Practical privacy: The sulq framework," in *ACM Principles of Database Systems (PODS)*, (Baltimore, Maryland), June 2005.

[11] Breese, J., Heckerman, D., and Kadie, C., ""Empirical Analysis of Predictive Algorithms for Collaborative Filtering"," in *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, July 1998.

[12] Canny, J., "Collaborative filtering with privacy," 2002.

[13] Canny, J., ""Collaborative Filtering with Privacy"," in *IEEE Symposium on Security and Privacy*, (Oakland, CA), pp. 45–57, May 2002.

[14] Canny, J., ""Collaborative Filtering with Privacy via Factor Analysis"," in *ACM SIGIR Conference on Research and Development in Information Retrieval*, (Tampere,Finland), pp. 238–245, Aug 2002.

[15] Cheeseman and Stutz, ""Bayesian Classification(Auto-Class): Theory and Results"," in *Advances in Knowledge Discovery and Data Mining* (Press, A., ed.), (Menlo Park, CA), pp. 153–180, 1995.

[16] Chien, Y.-H. and George, E. I., "A bayesian model for collaborative filtering," in *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, (San Francisco, California), Morgan Kaufmann, 1999.

[17] Cranor, L., Reagle, J., and Ackerman, M., "Beyond concern: Understanding net users attitudes about online privacy," 1999.

[18] Denning, D. and Schwartz, M., ""The Tracker A Threat to Statistical Database Security"," in *ACM Transactions on Database Systems*, vol. 4, pp. 76–96, 1979.

[19] Estivill-Castro, V. and Brankovic, L., ""Data Swapping: Balancing Privacy Against Mining of Association Rules"," in *Proc. of Knowledge Discovery and Data Warehousing*, (Florence, Italy), pp. 389–398, Aug 1999.

[20] Evfimievski, A., Gehrke, J., and Srikant, R., ""Limiting Privacy Breaches in Privacy Preserving Data Mining"," in *Principles of Database Systems*, (San Diego, CA), June 2003.

[21] Goldberg, D., Nichols, D., Oki, B., and Terry, D., ""Using Collaborative Filtering to Weave an Information Tapestry"," *Communications of the ACM*, vol. 35, pp. 61–70, Dec 1992.

[22] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C., "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[23] Gomatam, S. and Karr, A., ""Distortion Measures for Categorical Data Swapping"," Tech. Rep. 131, US National Institute for Statistical Sciences, Jan 2003.

[24] Gonzalez, R. and Woods, R., *"Digital Image Processing"*. Addison-Wesley Publishing Company, 1992.

[25] Grouplens. "http://www.grouplens.org/data/", May 20, 2005.

[26] Herlihy, M., "A methodology for implementing highly concurrent data objects," *ACM Trans. Program. Lang. Syst.*, vol. 15, pp. 745–770, November 1993.

[27] Herlocker, J., Konstan, J., Borchers, A., and Reidl, J., ""An Algorithmic Framework for Collaborative Filtering"," in *ACM SIGIR Conference on Research and Development in Information Retrieval*, (Tampere, Finland), Aug 2002.

[28] Herlocker, J., Konstan, J., Borchers, A., and Reidl, J., ""An Algorithmic Framework for Collaborative Filtering"," in *ACM SIGIR Conference on Research and Development in Information Retrieval*, (Tampere, Finland), Aug 2002.

[29] Herlocker, J., Konstan, J., and Riedl, J., "" Explaining Collaborative Filtering Recommendations"," in *ACM 2000 Conference on Computer Supported Cooperative Work*, pp. 241–250, Dec 2000.

[30] Hill, W., Stead, L., Rosenstein, M., and Fumas, G. W., “”recommending and evaluating choices in a virtual community of use”,” in *ACM Conference on human factos in computer systems CHI'95*, (Denver, Colorado), pp. 194–201, 1995.

[31] “http://www.almaden.ibm.com/software/quest/resources/ datasets/syndata.html,” Nov 05, 2004.

[32] Ishitani, L., Almeida, V., and Meiru, W., “”Masks: Bringing Anonimity and Personalization Together”,” in *Ninth INFORMS Conference on Information Systems and Technology*, Oct 2004.

[33] J.S, B., D., H., and C, K., “Emperical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, pp. 43–52, 1998.

[34] Klosgen, W., “”Anonimization Techniquesfor Knowledge Discovery in Databases”,” in *Proc. of the First International Conference on Knowledge and Discovery in Data Mining*, (Montreal, Canada), pp. 186–191, Aug 1995.

[35] Lam, S. and Riedl, J., “Shilling recommender systems for fun and profit.”

[36] Lamport, L., *LaTeX User's Guide and Document Reference Manual*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1986.

[37] Massa, P. and Avesani, P., “Trust-aware collaborative filtering for recommender systems,” 2004.

[38] Massa, P. and Bhattacharjee, B., “Using trust in recommender systems: an experimental analysis,” 2004.

[39] Mayerson, A. and Williams, R., “”On the Complexity of Optimal k-Anonimity”,” in *Proc. of the 23rd ACM-SIGMOD-SIGACT-SIGART Symposium on the Principles of Database Systems*, pp. 223–238, 2004.

[40] Moore, R., “”Controlled Data-swapping Techniques for Masking Public Use Microdata Sets”,” in *SRD Report RR 96-04, U.S. Bureau of the Census*, 1996.

[41] Moursmund, D., “”Chebyshev Solution of n+1 Linear Equations in n Unknowns”,” *Journal of the ACM*, vol. 12, pp. 383 – 387, July 1965.

[42] Muralidhar, K. and Sarathy, R., “”Disclosure Risk and Data Utility Characteristics of Data Swapping: A Preliminary Investigation,” in *IEEE Security and Privacy*, vol. 1, pp. 18–23, May 2003.

[43] http://mathworld.wolfram.com/Affine.html, June 20, 2004.

[44] Oliveira, S. and Zaane, O., “”Privacy Preserving Clustering by Data Transformation”,” in *Proc. of the 18th Brazilian Symposium on Databases*, (Manaus, Brazil), pp. 304–318, Oct 2003.

[45] Oliveira, S. and Zaane, O., “”Achieving Privacy Preservation When Sharing Data for Clustering”,” in *Workshop on Secure Data Management in conjunction with VLDB2004*, (Toronto, Canada), Springer Verlag LNCS 3178, Aug 2004.

[46] Parameswaran, R. and Blough, D., "A Robust Data-obfuscation Approach for Privacy Preservation of Clustered Data," in *Workshop Proceedings of the 2005 IEEE International Conference on Data Mining*, (Houston, Texas), pp. 18–25, IEEE, 2005.

[47] Parameswaran, R. and Blough, D., ""An Investigation of the Cluster Preservation Property of Nends"," tech. rep., Georgia Institute of Technology, 2005.

[48] Pennock, D., Horovitz, E., Lawrence, S., and Gilles, C., ""Collaborative Filtering by Personaliyt Diagnosis: A Hybrid Memory and Model Based Approach"," in *Sixteenth Conference on Uncertainty i Artificial Intelligence*, (San Francisco, CA), pp. 473–480, Morgan Kauffmann, 2000.

[49] Pennock, D., Horvitz, E., Lawrence, S., and Giles, C. L., "Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI 2000*, (Stanford, CA), pp. 473–480, 2000.

[50] Polat, H. and Du, W., "Privacy-preserving collaborative filtering using randomized perturbation techniques," 2003.

[51] Reiss, S. P., ""Practical Data-swapping The First Steps"," in *ACM Transactions on Database Systems*, vol. 9, pp. 20–37, Mar 1984.

[52] Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., and Riedl, J., "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," in *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, (Chapel Hill, North Carolina), pp. 175–186, ACM, 1994.

[53] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J., ""Grouplens: An Open Architecture for Collaborative Filetering of Netnews," in *ACM Conference on Computer Supported Cooperative Work*, pp. 175–186, 1994.

[54] Resnick, P. and Varian, H. R., "Recommender Systems," in *Communications of the ACM*, vol. 4, pp. 56–58, ACM, 1997.

[55] Rivest, R., Adleman, L., and Dertouzas, M., ""On Data Banks and Privacy Homomorphisms"," in *Foundations of Secure Computations* (et al, R. A. D., ed.), pp. 169–179, Academic Press, 1978.

[56] Rotenberg, M., ""The Privacy Sourcebook 2000: United States Law, International Law, and Recent Developments"." Electronic Privacy Information Center, 2000.

[57] Salas, S. and Hille, E., *Calculus: One and Several Variable*. New York: John Wiley and Sons, 1978.

[58] Samarati, P., ""Protecting Respondent's Privacy in Microdata Release"," *IEEE Transactions on Knowledge and Databases*, vol. 13, no. 6, 2001.

[59] Sandhu, R., Coyne, E., Feinstein, H., and Youman, C., "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[60] Shardanand, U. and Maes, P., "Social information filtering: Algorithms for automating "word of mouth"," in *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, vol. 1, pp. 210–217, 1995.

[61] Stallings, W., *"Network Security Essentials"*. Prentice Hall, 2000.

[62] Sweeney, L., ""k-Anonymity: A Model for Protecting Privacy"," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557–570, 2002.

[63] Toussaint, G., ""Bibliography on Estimation of Misclassification"," *IEEE Transactions on Information Theory*, vol. 20, pp. 472–479, July 1974.

[64] http://kdd.ics.uci.edu/, Oct 21, 2004.

[65] Ungar, L. and Foster, D., "Clustering methods for collaborative filtering," in *Proceedings of the Workshop on Recommendation Systems*, AAAI Press, Menlo Park California, 1998.

[66] Upendra, S., "Social information filtering for music recommendation," 1994.

[67] Warrior, J., McHenry, E., and McGee, K., ""They Know Where You Are"," *IEEE Spectrum*, vol. 40, no. 7, pp. 20–25, 2003.

[68] Winkler, W., ""Using Simulated Annealing for k-Anonymity"," in *Research Report Series, U.S. Census Bureau*, 2002.

[69] Wolfgang, C. D., "Preventing shilling attacks in online recommender systems paul-alexandru chirita."

[70] Ziegler, C.-N. and Freiburg, D. "http://www.informatik.uni-freiburg.de/ cziegler/BX/", May 15, 2005.

# INDEX