K-ANONYMITY USING GRAPH THEORY

PROJECT WORK

Submitted

in partial fulfillment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

in the faculty of

COMPUTER SCIENCE & ENGINEERING

by

GUDIWADA RAGHAVA

[R. No. 14021A0531]

Under the Guidance of

Mrs. KARUNA ARAVA

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

MAY 2018

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

MAY 2018

DECLARATION

I hereby declare that the project work described in this thesis, entitled ***"K-Anonymity Using Graph Theory"***, which is being submitted by me in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology (B.Tech.)*** in the faculty of Computer Science & Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada – 533003, A.P., is the result of investigations carried out by me under the guidance of Mrs. Karuna Arava, Assistant Professor of Computer Science & Engineering, University College of Engineering, Jawaharlal Nehru Technological University Kakinada - 533003.

The work is original and has not been submitted to any other University or Institute for the award of any degree or diploma.

Place: Kakinada **Signature:**

Date: **Name: GUDIWADA RAGHAVA**

Roll No.: 14021A0521

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (AUTONOMOUS)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533003, A.P., INDIA

CERTIFICATE

This is to certify that the project work entitled *“**K-Anonymity Using Graph Theory**”* that is being submitted by Mr. Mazindrani Mohmmed Jawad, Roll No. 14021A0521, in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology (B.Tech.)*** in the faculty of Computer Science & Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada – 533003, A.P. is a record of bonafide project work carried out by him under my guidance and supervision.

The results embodied in this project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of Project Work Supervisor

(Mrs. Karuna Arava)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY

KAKINADA – 533 003, A.P., INDIA

CERTIFICATE

This is to certify that the project work entitled ***"K-Anonymity Using Graph Theory"*** that is being submitted by Mr. Mazindrani Mohmmed Jawad, Roll No. 14021A0521, in partial fulfillment of the requirements for the award of degree of ***Bachelor of Technology (B.Tech.)*** in the faculty of Computer Science & Engineering to the University College of Engineering (Autonomous), Jawaharlal Nehru Technological University Kakinada – 533003, A.P. is a record of bonafide project work carried out by him at our department.

Signature of Head of the Department

Dr. K. Sahadevaiah,

Professor of CSE & Head

ACKNOWLEDGEMENTS

I express my deep gratitude and regards to Karuna Arava, internal guide and assistant professor, Department of Computer Science and Engineering, University College of Engineering (Autonomous), Jawaharlal Nehru Technological University, Kakinada for her encouragement and valuable guidance in bringing shape to this dissertation.

I am grateful to Dr. K. Sahadevaiah, Head of the Department of Computer Science and Engineering, University College of Engineering (Autonomous), Jawaharlal Nehru Technological University, Kakinada for his encouragement and motivation.

I am thankful to all the Professors and Faculty Members in the department of their teachings and academic support and thanks to Technical Staff and Non-Teaching staff in the department for their support.

GUDIWADA RAGHAVA

Roll No : 14021A0531

Abstract

K-anonymity is a model to protect public released microdata from individual identification. It requires that each record is identical to at least k−1 other records in the anonymized dataset with respect to a set of privacy-related attributes. Although it is easy to anonymize the original dataset to satisfy the requirement of k-anonymity, it is important to ensure that the anonymized dataset should preserve as much information as possible of the original dataset. To minimize the information loss due to anonymization, it is crucial to group similar data together and then anonymize each group individually.

This work is based on graph theory. This is a greedy algorithm that considers the actual table to be a graph with nodes as records and edges as information loss between the nodes. This algorithm minimizes the information loss by selecting the edges with lowest weights possible. It is scalable for multi-dimensional tables compared with existing techniques in the literature.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

T	No. of records
N1, . . . , Nm	Numeric quasi identifiers
C1, . . . , Cq	Categorical quasi identifiers
N ˆi(P)	Maximum value of the records in P.
N ˇi(P)	Minimum value of the records in P.
N i(P)	Average value of the records in P.

CHAPTER 1

INTRODUCTION

1.1 What is Cloud?

The term Cloud refers to a Network or Internet. In other words, we can say that Cloud is something, which is present at remote location. Cloud can provide services over network, i.e., on public networks or on private networks, i.e., WAN, LAN or VPN. Applications such as e-mail, web conferencing, customer relationship management (CRM), all run in cloud.

Clouds are owned by large single parties such as individual companies and are usually used by small to medium commercial businesses and researchers. These users normally pay the provider to use their computing resources.

1.2 What is Cloud Computing?

The name cloud computing came from the cloud symbol that is frequently used to represent the Internet in diagrams and flowcharts. Cloud Computing refers to manipulating, configuring, and accessing the applications online. It offers online data storage, infrastructure and application.

Today cloud computing covers anything that involves delivering hosted services over the internet. These services are generally divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).

All cloud services have three unique characteristics that distinguish them from traditional hosting:

- They are commercial and sold on demand to user
- They are flexible - a user can have as little or as much of the service as he wants

at any time and can quickly outsource peaks of activity without long term commitment

- They are fully managed by the cloud provider; the consumer requires only a computer and internet access.

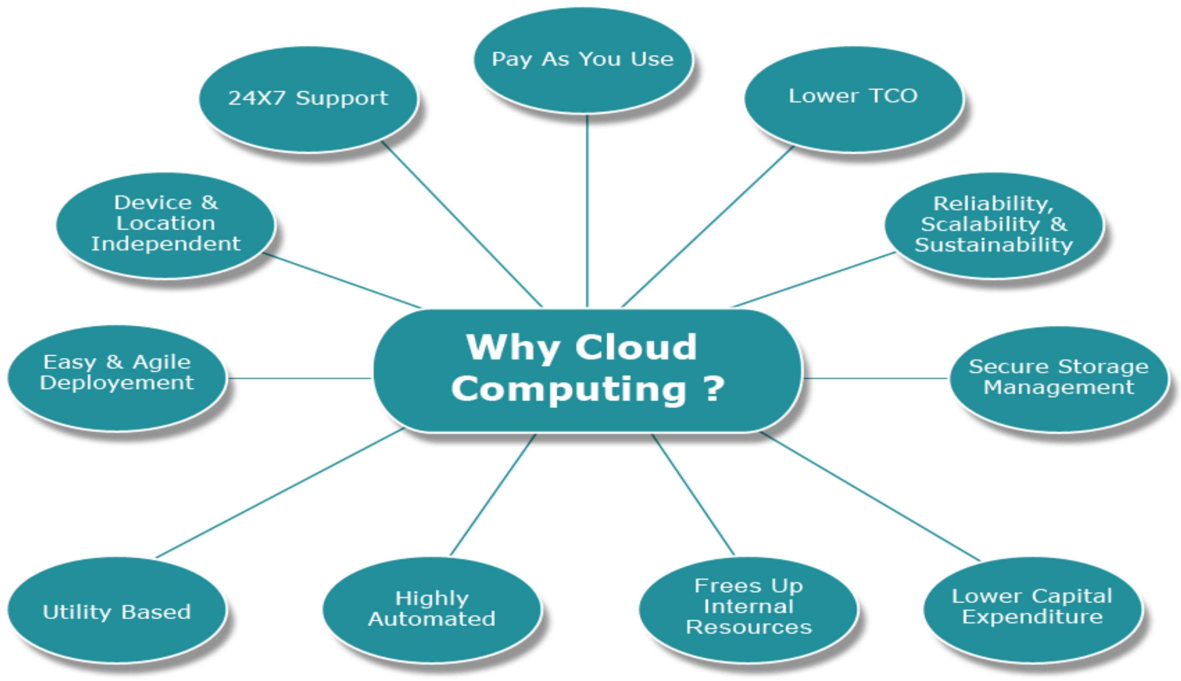

Why Cloud Computing ?
Pay As You Use
24X7 Support
Lower TCO
Device & Location Independent
Reliability, Scalability & Sustainability
Easy & Agile Deployement
Secure Storage Management
Utility Based
Highly Automated
Frees Up Internal Resources
Lower Capital Expenditure

Fig 1.1 Overview of Cloud Computing

1.3 How do clouds work?

One of the benefits of clouds is their relative ease of use. When a user needs extra computing services they simply pick the right cloud for their needs, pay for what they've used and discard it when they're done.

Clouds often employ a technique called 'virtualization' to give users more control when using the cloud. Virtualization allows users to create their own 'virtual computer', with specified applications, software and operating machines. For example we could set up a Windows-based virtual machine but run it inside a Mac.

Virtualization means that users don't need to worry about what software is being run where. This is especially important with clouds, as they won't often know on which machines their data or work will be running. However, on the down side, setting up virtual machines is quite difficult to do!

1.4 Types of Service Models

Clouds can be both private and public. Public clouds sell their services to anyone on the internet (e.g. Amazon Web services). Private clouds, on the other hand, are proprietary networks or data centres that supply hosted services to a selected number of people – e.g. within a single company.

Whether private or public, the aim of cloud computing is to provide easy access to computing resources and IT services.

Clouds can also be classified according to the different services they offer:

- **Infrastructure-as-a-Service (IaaS)** - buying access to computing capacity over the internet, such as servers or storage. Also known as utility computing, as it is like buying in a utility service such as gas or electricity. IaaS often employs virtualization so users can create their own "virtual computer". This means they can specify the applications, software and operating system they want to deploy in the cloud. Advantages are that users don't need to worry about purchasing and running their own hardware but disadvantages include that it can be difficult to package up and run a computer remotely.
- **Platform-as-a-service (PaaS)** - developing applications that use web-based tools, so they run in a software environment (i.e. a platform) provided by another company. Platform as a service (PaaS) or application platform as a service (aPaaS) is a category of services that provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. PaaS can be delivered in two ways: as a public cloud service from a provider, where the consumer controls software deployment with minimal configuration options, and the provider provides the

networks, operating systems (OS), 'middleware' (e.g. Java runtime, .NET runtime, integration, etc.), database other services to host the consumer's application; or as a private service (software) inside the firewall, or as software deployed on a public infrastructure as a service.

- **Software-as-a-service (SaaS)** - SaaS is a software delivery in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as "on-demand software", and was formerly referred to as "software plus services" by Microsoft. SaaS is typically accessed by users using a thin client via a web browser. SaaS has become a common delivery model for many business applications. SaaS has been incorporated into the strategy of nearly all leading enterprise software companies.

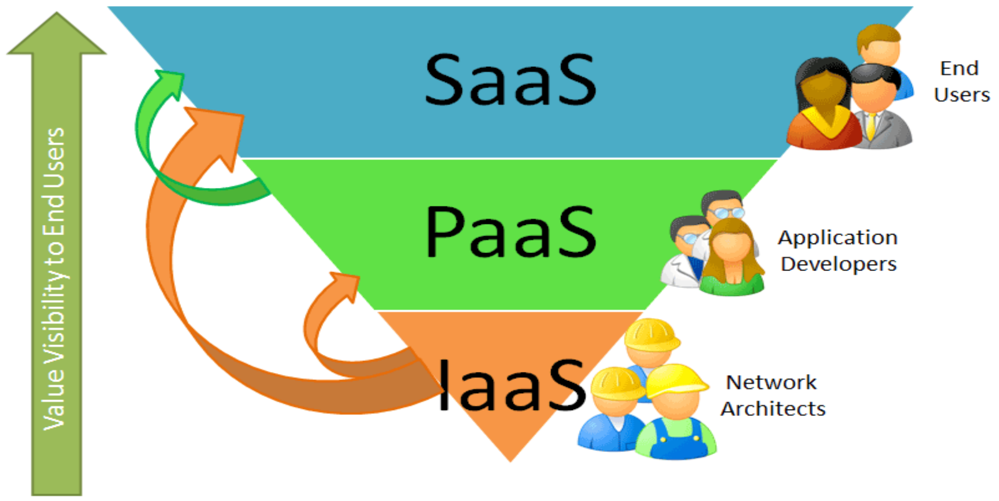


Fig.1.2 Cloud Service Models

1.5 Deployment Models

There are four different deployment models of cloud computing:

- **Public Cloud:** Public or external cloud is traditional cloud computing where resources are dynamically provisioned on a fine-grained, self-service basis over the Internet or via and or from an off-site third-party provider who bills on a fine-grained basis
- **Community Cloud:** If several organizations have similar requirements and seek to share infrastructure to realize the benefits of cloud computing, then a community cloud can be established. This is a more expensive option as compared to public cloud as the costs are spread over fewer users as compared to a public cloud. However, this option may offer a higher level of privacy, security and/or policy compliance.
- **Hybrid Cloud:** Hybrid Cloud means either two separate clouds joined together (public, private, internal or external) or a combination of virtualized cloud server instances used together with real physical hardware. The most correct definition of the term "Hybrid Cloud" is probably the use of physical hardware and virtualized cloud server instances together to provide a single common service. Two clouds that have been joined together are more correctly called a "combined cloud".
- **Private Cloud:** Private clouds describe offerings that deploy cloud computing on private networks. It consists of applications or virtual machines in a company's own set of hosts. They provide the benefits of utility computing - shared hardware costs, the ability to recover from failure, and the ability to scale up or down depending upon demand.

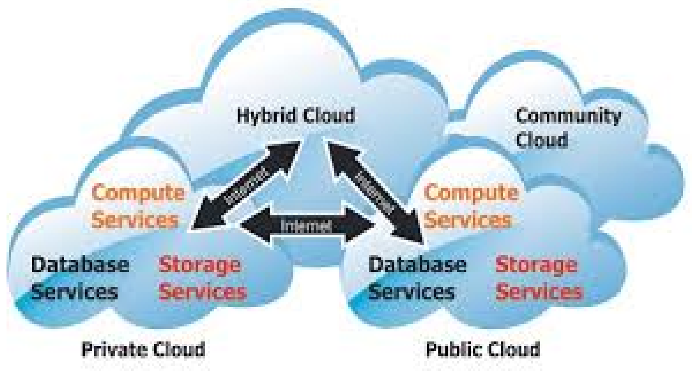


Fig 1.3 Deployment Models

Privacy protection is an important issue in today's society. As the public is more concerned with privacy, protection of public released micro data from individual identification becomes even more important. Most privacy protection techniques rely on randomization [1] or generalization [2] of the original data. However, such techniques can also damage the quality of the original data. Therefore, this poses a dilemma between data quality and data privacy. The k-anonymity model [2][3] is a privacy-preserving approach that has been extensively studied for the past few years. It works by ensuring that each record of a table is,

Zip Code	Gender	Age	Disease	Expense
75275	Male	22	Flu	100
75277	Male	23	Cancer	3000
75278	Male	24	Hiv	5000
75275	Male	33	Diabetis	1000

Table 1.1: General Data Records

identical to at least k−1 other records with respect to a set of privacy-related attributes, called quasi-identifiers, that could be potentially used to identify individuals by linking these attributes to external data sets. For example, consider the hospital data in Table 1, where the attributes Zip Code, Gender and Age are regarded as quasi-identifiers. Table 2 gives a 3-anonymization version of Table 1, where anonymization is achieved via generalization at the attribute level [4], i.e., if two records contain the same value at a quasi-identifier, then they are generalized to the same value at the quasi-identifier as well. Table 3 gives another 3- anonymization version of Table 1, where anonymization is achieved via generalization at the cell level [4], i.e., two cells with same value could be generalized to different values (e.g., value 75275 in the Zip Code column and value Male in the Gender column). Since anonymization via generalization at the cell level generates data containing different

Zip Code	Gender	Age	Disease	Expense
7527*	Person	21-30	Flu	100
7527*	Person	21-30	Cancer	3000
7527*	Person	21-30	Hiv	5000
7527*	Person	31-40	Diabetes	1000

Table 1.2: Anonymizations at Attribute Level

Zip Code	Gender	Age	Disease	Expense
7527*	Male	21-25	Flu	100
7527*	Male	21-25	Cancer	3000
7527*	Male	21-25	Hiv	5000
75275	Person	31-40	Diabetes	1000

Table 1.3: anonymization at Cell Level

Generalization levels within a column, utilizing such data becomes more complicated than utilizing the data generated via generalization at the attribute level. However, in term of data quality, generalization at the cell level incurs less information loss than generalization at the attribute level, and therefore is adopted in this study.

1.6 Problem Statement

K-anonymity is a model requires that each record is identical to at least k − 1 other records in the anonymized dataset with respect to a set of privacy related attributes. Although it is easy to anonymize the original dataset to satisfy the requirement of k-anonymity, it is important to ensure that the anonymized dataset should preserve as much information as possible of the original dataset. To minimize the information loss due to anonymization, it is crucial to group similar data together and then anonymize each group individually. In this work compares the performance of two recently proposed clustering-based techniques for k-anonymization, and proposes a hybrid of both techniques to achieve less information loss than each of the original techniques.

1.7 Scope

Most privacy protection techniques rely on randomization [1] or generalization [2] of the original data. However, such techniques can also damage the quality of the original data. Therefore, this poses a dilemma between data quality and data privacy. So the anonymization technique which provide the quality data without damaging the privacy is needed in this real world. This generalization method using graph theory is one of that novel method.

1.8 Objective

Our objective is providing security to data without losing its privacy and also not damaging the utility (ease of use) nature of that data.

1.9 Proposed Approach and method to be employed

The basic ideas of this method is to represent the records as nodes of a graph. These nodes are connected with weighted edges that represent the amount of information loss will occur when the two nodes are combined into one. The algorithm works in two steps. In the first step nodes that are near to each other are combined into merged nodes. In the second step the nodes are redistributed such that each merged node has at least k nodes in it.

1.10 Organization of Thesis

In the Chapter 1, an introduction to the cloud and the privacy problems involved are discussed. Light is thrown on literature survey in the Chapter 2. Chapter 3 comprises the system design and the design diagrams. A brief description of work is shown in Chapter 4 which is followed with its results in the Chapter 5. Chapter 6 gives the testing strategies for the method. Finally conclusion is presented in Chapter 7.

CHAPTER 2

LITERATURE SURVEY

2.1 Recommended System Strategies

The k-anonymity model has attracted much attention for the past few years. Many approaches have been proposed for k-anonymization and its variations. Please refer to Ciriani et al. [4] for a survey of various k-anonymization approaches. This section first describes the concept of information loss, which will be used throughout this paper to evaluate the effectiveness of k-anonymization approaches. The notion of information loss is used to quantify the amount of information that is lost due to k-anonymization. The description in this subsection is based on Byun et al [5].

Let T denote a set of records, which is described by m numeric quasi-identifiers $N_1, \ldots, N_m$ and q categorical quasi identifiers $C_1, \ldots, C_q$. Let $P = \{P_1, \ldots, P_p\}$ be a partitioning of T , namely, $\cup_{i\in[1,p]} P_i = T$, and $P_{\hat{i}} \cap P_{\check{i}} = \emptyset$ for any $\hat{i} \neq \check{i}$. Each categorical attribute C_i is associated with a taxonomy tree T_{Ci} , that is used to generalize the values of this attribute. Consider a set $P \subset T$ of records. Let $\hat{N}_i(P)$, $\check{N}_i(P)$ and $N_i(P)$ respectively denote the max, min and average values of the records in P with respect to the numeric attribute N_i. Also, let $\check{C}_i(P)$ denote the set of values of the records in P with respect to the categorical attribute C_i, and let $T_{Ci}(P)$ denote the maximal subtree of T_{Ci} rooted at the lowest common ancestor of the values of $\check{C}_i(P)$. Then, the diversity of P, denoted by D(P), is defined as:

$$D(P) = \sum_{i\in[1,m]} \hat{N}_i(P) - \check{N}_i(P) / \hat{N}_i(T) - \check{N}_i(T) + \sum_{i\in[1,q]} H(T_{Ci}(P)) / H(T_{Ci})$$

where H(T) represents the height of tree T.

Let r_0 and r_* be two records, then the distance between r_0 and r_* is defined as the diversity of the set $\{r_0, r_*\}$, i.e., $D(\{r_0, r_*\})$. The centroid of P is a record whose value of attribute N_i equals $N_i(P)$, and value of attribute C_i equals the lowest common ancestor of the values of $\check{C}_i(P)$ in the taxonomy tree T_{Ci} . The distance between a record r and a cluster P is defined as the number of records in P times the distance between r and the centroid of P.

To anonymize the records in P means to generalize these records to the same values with respect to each quasi-identifier. The amount of information loss occurred by such a process, denoted as L(P), is defined as:

$$L(P) = |P| \times D(P).$$

Where $|P|$ represents the number of records in P.

Consider a partitioning $P = \{P_1, \ldots, P_p\}$ of T . The total information loss of P is the sum of the information loss of each $P_{i\in[1,p]}$. To ensure the data quality of T after anonymization, an anonymization method should try to construct a partitioning P such that total information loss of P is minimized.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

An overview of system architecture was discussed in this Chapter. UML diagrams and technology used also discussed.

3.1 System Architecture

Following figure shows the flow of data and how user input gets processed by the system. The major building blocks of the architecture are data owners and the key management units along with cloud service provider. As the diagram shoes that all the users are keeping their records in the cloud with a key used to encrypt their data records. The key manager unit authenticate the users to get into the cloud for accessing the records. The users can access the records only after authentication [3].

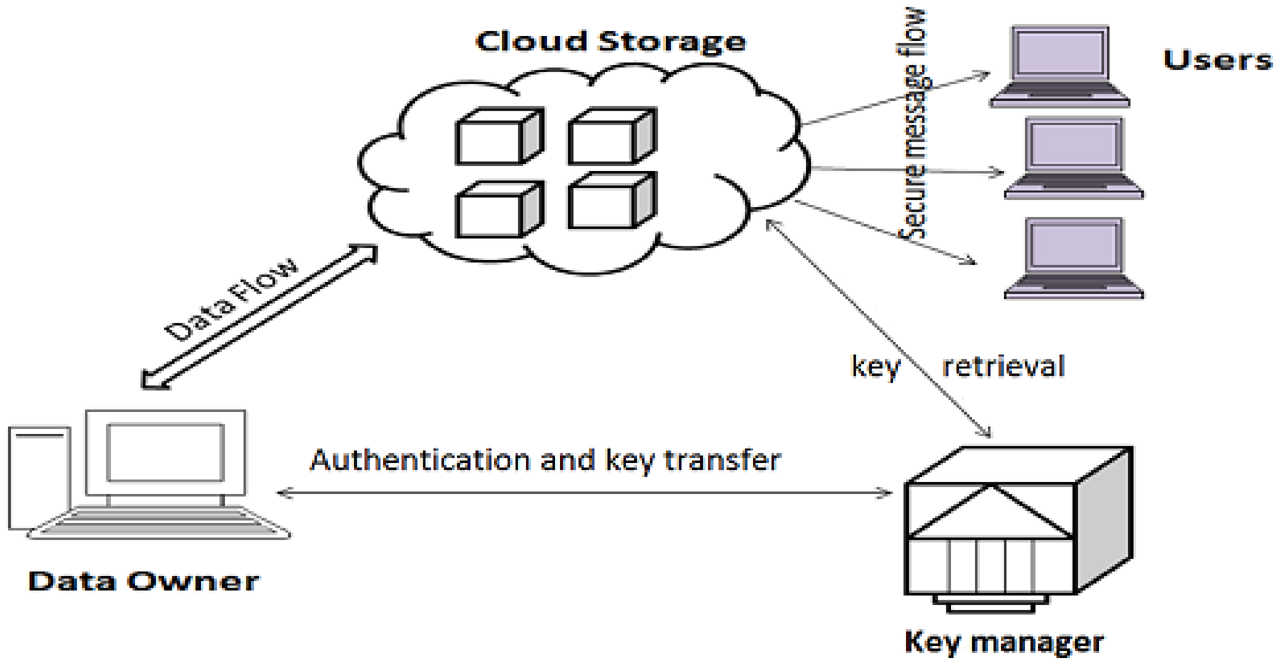


Fig. 3.1: System Architecture

3.2 System Design

The different diagrams of the UML are discussed below.

3.2.1 Use Case Diagram

Use case is a list of steps, typically defining interactions between a role (known in UML as an "actor") and a system, to achieve a goal. The actor can be a human or an external system.

In the Unified Modeling Language, the relationships between all (or a set of) the use cases and actors are represented in a Use Case Diagram or diagrams.

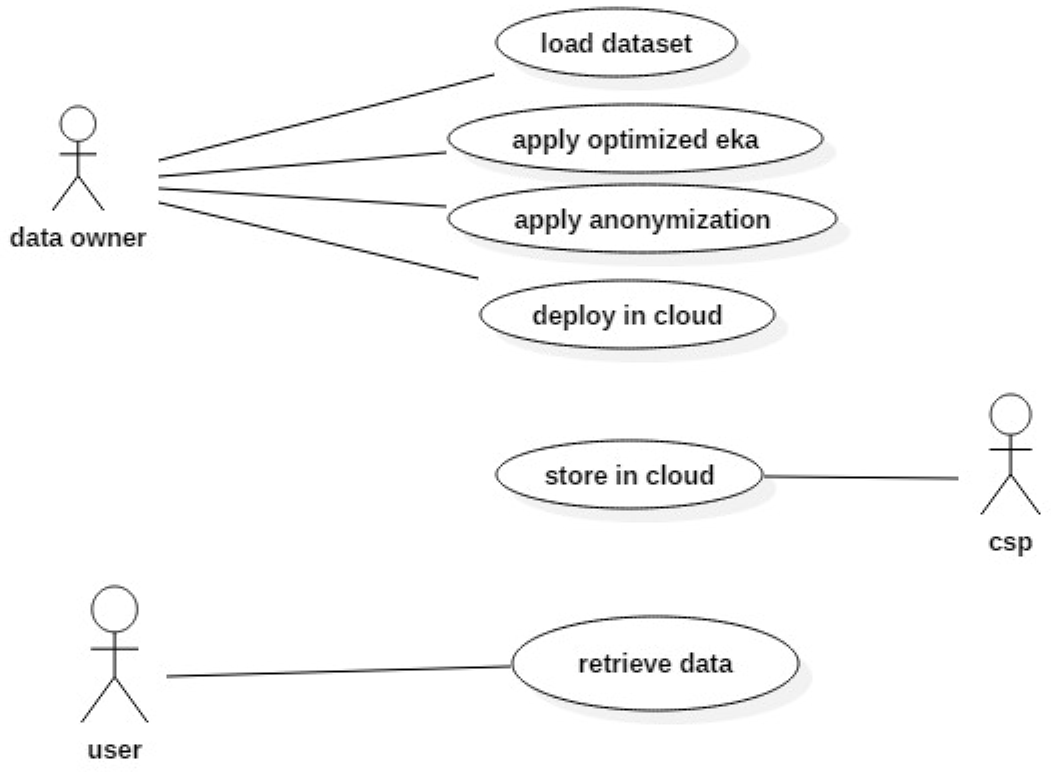


Fig. 3.2: Use Case Diagram

3.2.2 Sequence Diagram

A Sequence diagram in a Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. They can be useful to check the development time, members needed in a team and also cost needed for the development of the system.

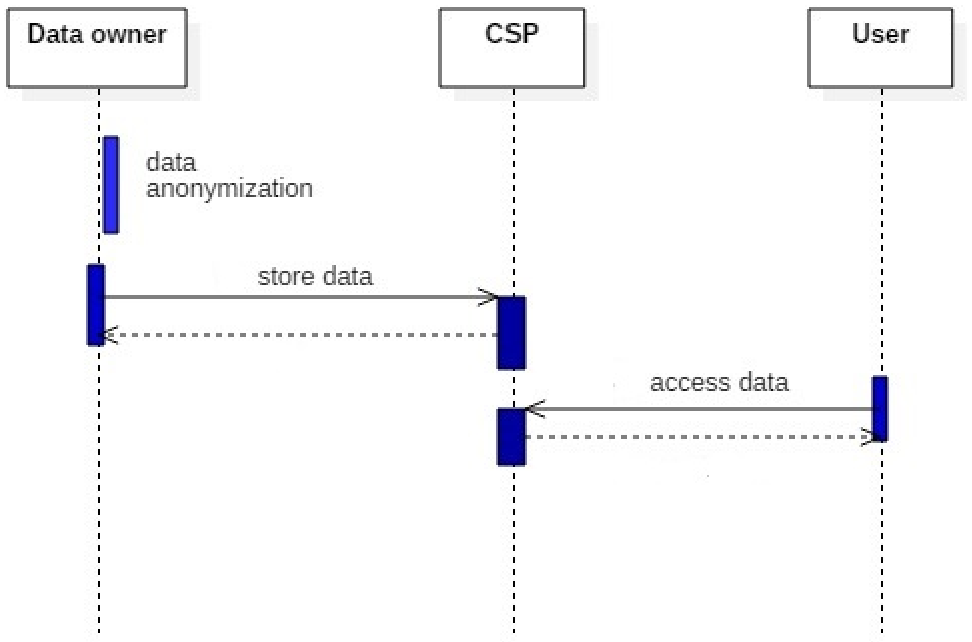


Fig. 3.3: Sequence Diagram

3.2.3 State Chart Diagram

A State chart diagram in unified modelling language is a type of dynamic structure diagram that describes the system flow of events by showing the major classes in terms of exchange of messages from time to time. State chart diagram descript the flow of steps that a particular state will undergo during implementation of any service.

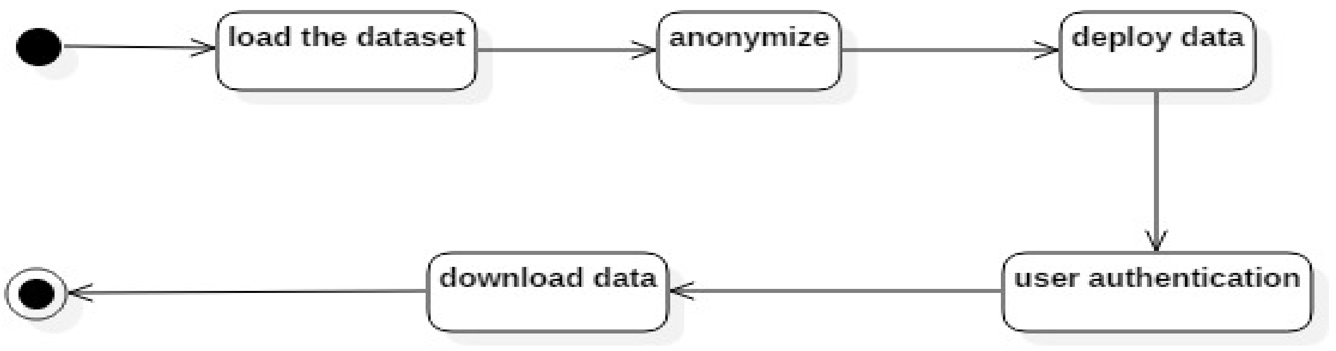


Fig. 3.4: State Chart Diagram

3.2.4 Class Diagram

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

A class diagram shows a set of classes, interfaces, associations and generalizations. Package is commonly used model element for organizing elements in class diagram. Class diagrams are not just for visualizing and documenting structure models but also for constructing executable system with forward, reverse and round-trip engineering.

The class diagram can be used to model the whole system at once. Different classes of the system can be further divided into other granular class diagrams, but the overall modeling of the system is shown by Level 0 class diagram as shown below.

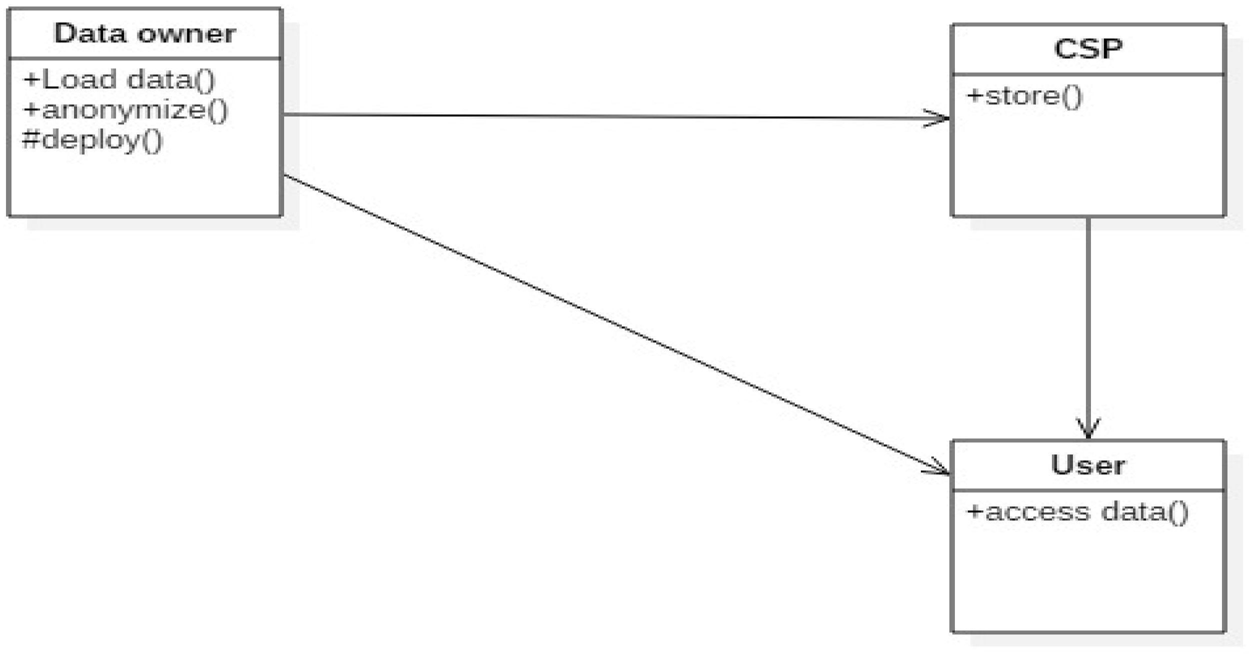


Fig. 3.5: Class Diagram

3.3 Technology Overview

The different technologies used for the computations are discussed below.

3.3.1 Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language. Python is a scripting language like PHP, Perl, Ruby and so much more. It can be used for web programming.

Following are the features of the Python language.

Interpreted:

A program written in a compiled language like C or C++ is translated from the source language i.e. C/C++ into a language spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software just stores the binary code in the computer's memory and starts executing from the first instruction in the program. When you use an interpreted language like Python, there is no separate compilation and execution steps. You just *run* the program from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your specific computer and then runs it. All this makes using Python so much easier. You just *run* your programs - you never have to worry about linking and loading with libraries, etc. They are also more portable this way because you can just copy your Python program into another system of any kind and it just works!

Object Oriented:

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simple way of doing object-oriented programming, especially, when compared to languages like C++ or Java.

Extensible:

If you need a critical piece of code to run very fast, you can achieve this by writing that piece of code in C, and then combine that with your Python program.

Embeddable:

You can embed Python within your C/C++ program to give scripting capabilities for your program's users.

Extensive Libraries:

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI(graphical user interfaces) using Tk, and also other system-dependent stuff. wherever Python is installed.

3.3.2 Intellij IDE Environment

Features of this platform are

Coding assistance

The IDE provides certain features like code completion by analyzing the context, code navigation which allows jumping to a class or declaration in the code directly, code refactoring and options to fix inconsistencies via suggestions.

Built in tools and integration

The IDE provides integration with build/packaging tools like grunt, bower, gradle, and SBT. It supports version control systems like Git, Mercurial, Perforce, and SVN. Databases like Microsoft SQL Server, ORACLE, Postgre SQL, and My SQL can be accessed directly from the IDE.

Plugin ecosystem

IntelliJ supports plugins through which one can add additional functionality to the IDE. One can download and install plugins either from IntelliJ's plugin repository website or through the IDE's inbuilt plugin search and install feature. Currently IntelliJ IDEA Community edition has 1495 plugins available, where as the Ultimate edition has 1626 plugins available.

Supported languages

The Community and Ultimate editions differ in their support for various programming languages as shown in the following table.

CHAPTER 4

DESCRIPTION OF WORK

An overview of the paper was discussed in this Chapter.

4.1 Basic Metrics and Terms

4.1.1 Information Loss

The notion of information loss is used to quantify the amount of information that is lost due to k-anonymization. The description in this subsection is based on Byun et al [5]. Let T denote a set of records, which is described by m numeric quasi- identifiers $N_1, \ldots, N_m$ and q categorical quasi- identifiers $C_1, \ldots, C_q$. Let $P = \{P_1, \ldots, P_p\}$ be a partitioning of T , namely, $\cup_{i\in[1,p]} P_i = T$, and $P_{\hat{i}} \cap P_{\check{i}} = \emptyset$ for any $\hat{i} \neq \check{i}$. Each categorical attribute $C_{i\in[1,q]}$ is associated with a taxonomy tree TCi, that is used to generalize the values of this attribute.

The amount of information loss incurred by anonymization on P is defined as:

$$L(P) = |P| \times D(P).$$

4.1.2 NCP Metric Calculations

NCP calculation can be done in various ways. They are

1. Linear Regression
2. Sample Diversity Factor Method
3. Mean per Unit

4.2 K-Anonymity Using Graph Theory

4.2.1 The Idea And Pseudo Code

This algorithm considers the records in the table as nodes of a graph. The nodes are connected with weighted edges denoting the amount of information loss when two nodes are combined to form a merged node. The algorithm consists of two stages.

In the first step n/k random seed vertices are chosen . Every non seed vertex is iterated and placed into the group that is best for it with minimum amount of information loss. After first step the output would be n/k groups or merged nodes.

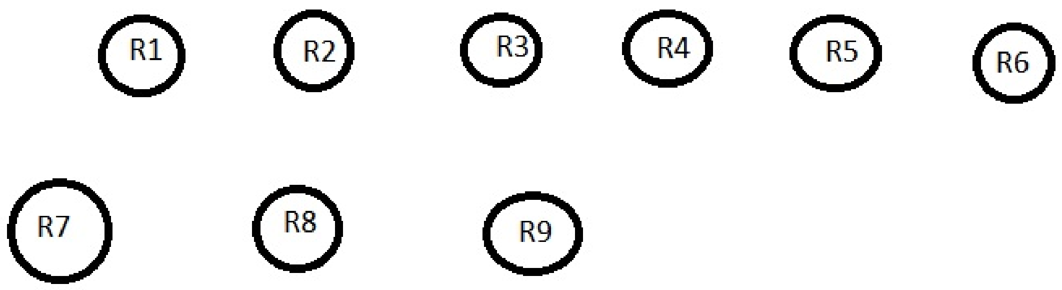
In the second step the records in the groups are redistributed such that each group has no less than k records so that the k-anonymity condition would be satisfied. This is done through removing farthest records from groups having more than k records and moving those records to groups having less than k records. The pseudocode is given below

```
Input: a partitioning P = {P1, . . . , Pp} of T
Output: an adjusted partitioning P = {P1, . . . , Pp} of T
1. Let R := ∅;
2. For each group P ∈ P do
3. if |P | > k then
4. Sort vertices in P by distance to P ’s centroid;
5. While (|P | > k) do
6. r ∈ P is the vertex farthest from P ’s centroid;
7. Let P := P \ {r}; R := R ∪ {r};
8. End of While
9. else if |P | < k then
10. P = P \ {P };
11. R = R ∪ P ;
12. End if
13. End of For
14. P = P ∪ R;
```

The output is made into k-anonymized ordering of records of T done through generalization.

A graphical representation of the steps is shown in the below images in a stepwise manner.

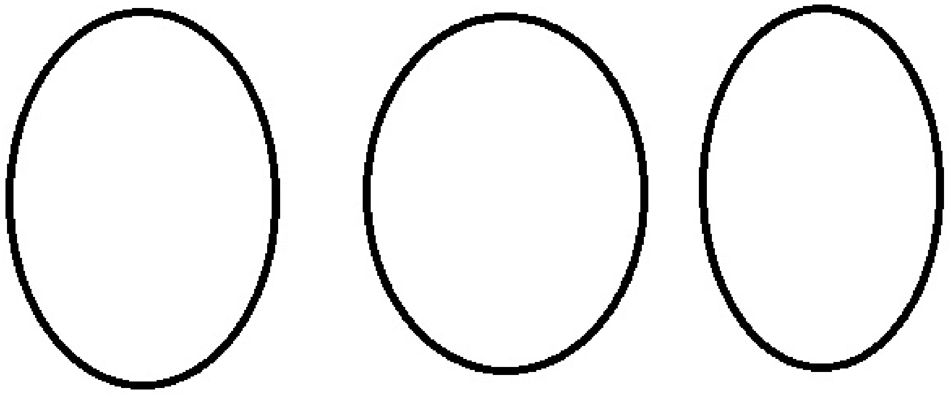
STEP 1: Consider each record as vertex. Let there be 9 records



STEP 2:

Here no.of records n=9 and k=3

Therefore n/k=9/3=3 groups are formed

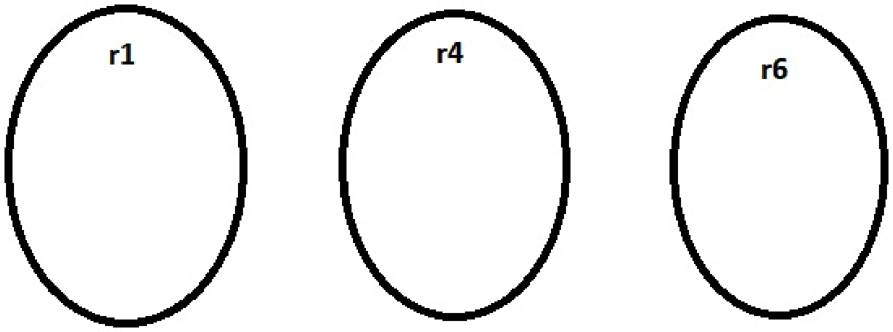


Step 3:

We randomly select n/k=3 records to place them in seed_groups

Let them be r1, r4, r6

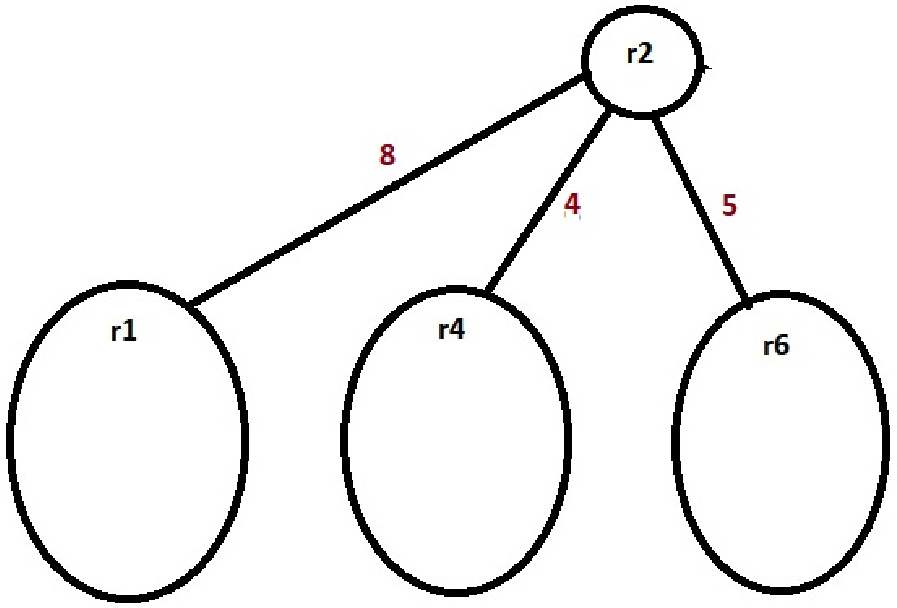
Add them to seed_groups



STEP 4 :

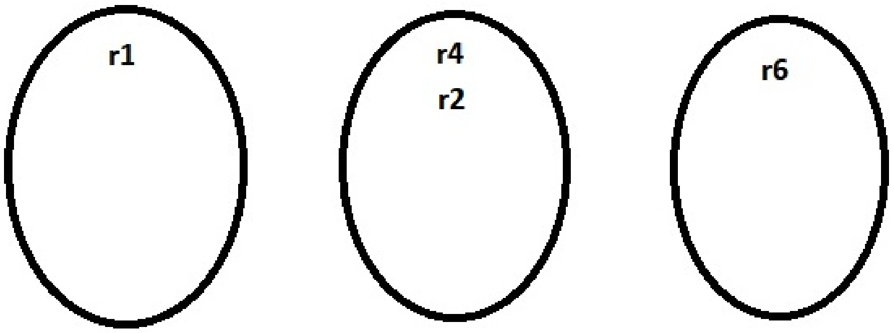
For each record not included in seed_groups find the best for it to place it and keep it in that

Since distance between group 2 and r2 is smallest r2 will be added to second group



STEP 5:

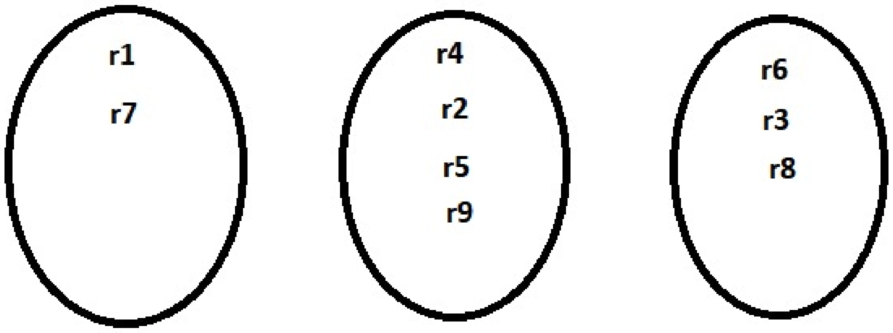
After addition of r2 to group 2 the image looks as follows



Do the same for all the records and add to the groups best for them

STEP 6:

After its completion the graph will be as follows



We can see that some groups have more than k records while others have less than k. So we call the adjust_group method

STEP 7:

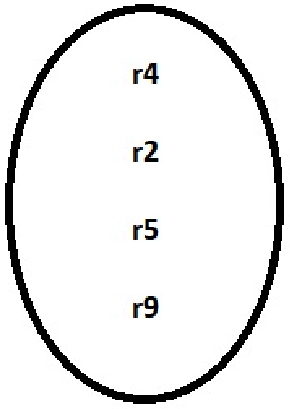
The adjust_group() method checks the information loss when any record is removed from it

Distance between r4 and generalized record of the group = 14

Distance between r2 and generalized record of the group = 20

Distance between r5 and generalized record of the group = 16

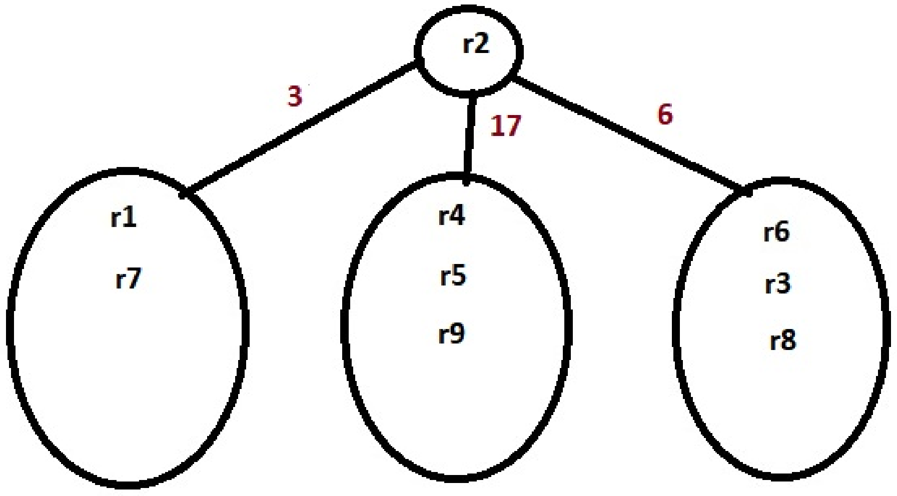
Distance between r9 and generalized record of the group = 8



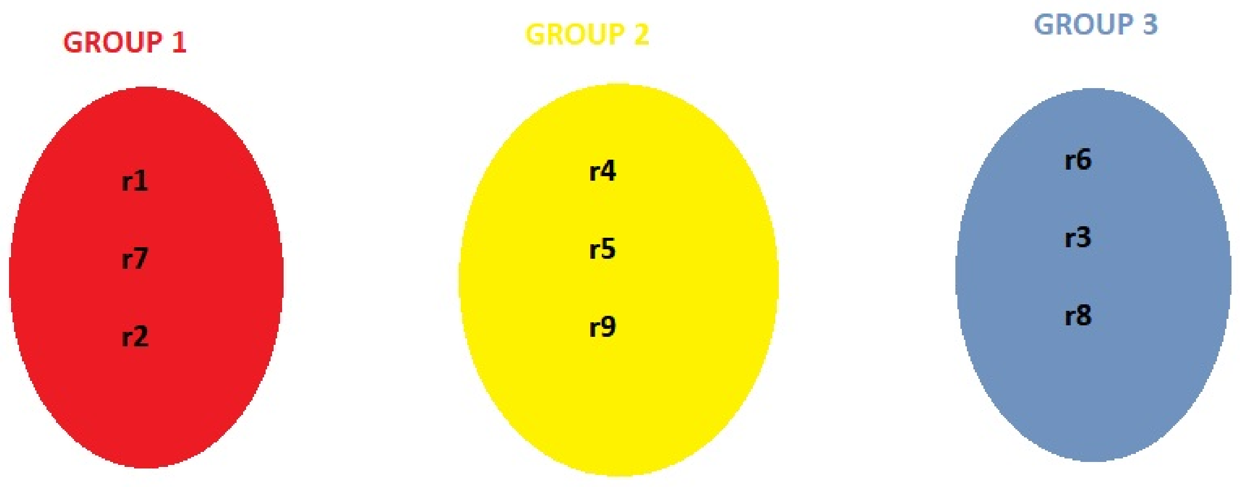
Hence r2 is removed since it is the farthest record from generalized record of the group

STEP 8:

The residual vertices are add to groups that are best fit for them
So we add r2 to group 1



The finalized k-anonymized graph look as follows:



4.2.2 Distance Metrics Used

Calculating the distance is done differently for different king of attributes. The attributes can be of two kinds namely, numeric and categorical. If the attributes are numeric then we calculate the distance as follows.

Let A be a numeric attribute with finite numeric attribute domain, N. If υ is the value of attribute A then the distance between two different records' values υ_o and υ_a is defined as:

$$\mathbf{Dist(\upsilon_o, \upsilon_{a)}} = | \upsilon_o - \upsilon_a | \; / N$$

where $N = \upsilon_{max} - \upsilon_{min}$

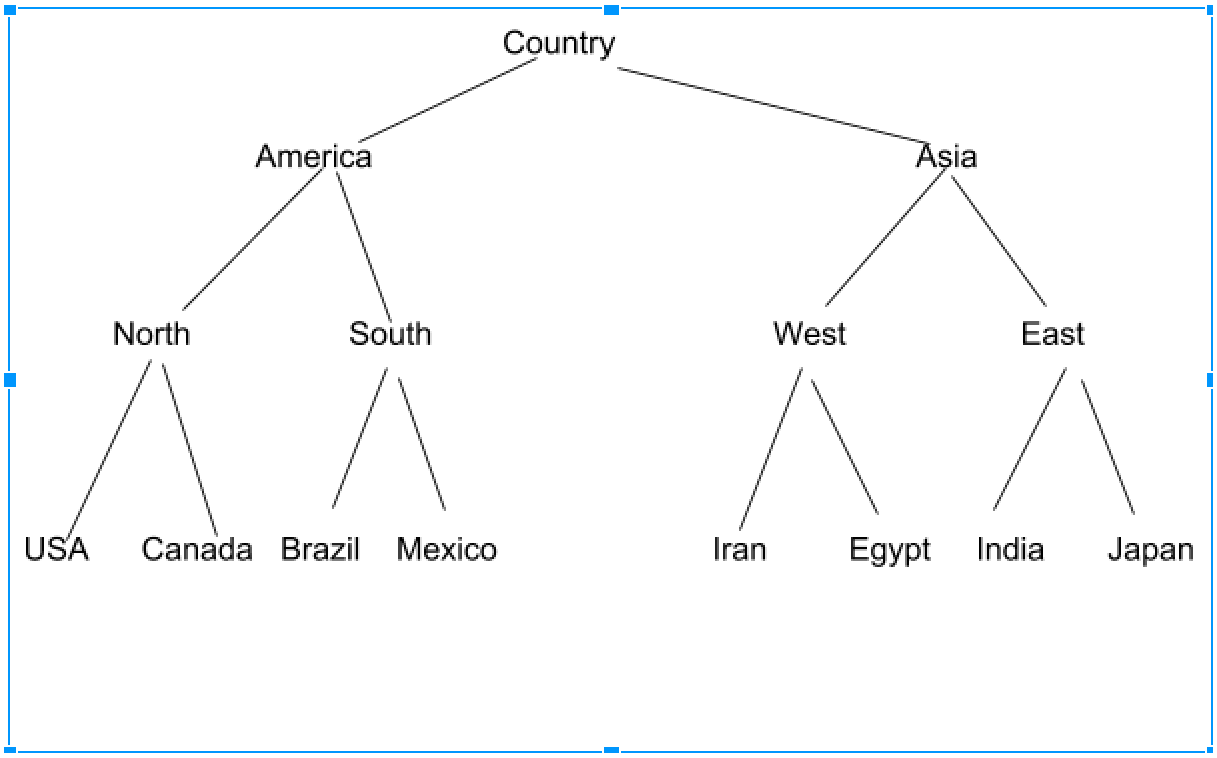
If the attribute is categorical then we draw the taxonomy tree of the attribute and calculate the distance between any two values of that categorical attribute using the below said metric.

Let D be a categorical domain and T_D be a taxonomy tree defined for D. The normalized distance between two values $v_i, v_j \in D$ is defined as:

$$\mathbf{Dist(\upsilon_o, \upsilon_{a)}} = H(\Lambda(v_i, v_j)) \; / \; H(T_D),$$

where $\Lambda(x, y)$ is the subtree rooted at the lowest common ancestor of x and y, and H(R) represents the height of tree T .

The below picture shows the taxonomy tree of categorical attribute country



For example let us calculate the distance between North and Mexico. It is calculated as

$$\begin{aligned} Dist(\text{North, Mexico}) &= H(\Lambda(\text{North, Mexico})) / H(\text{Tree}) \\ &= H(\text{America})/H(\text{country}) \\ &= 3/4 \\ &= 0.75 \end{aligned}$$

4.2.3 Distance Between Two Records

Let a record t_1 with values v_i , c_i where i=1,2,3,.....,N is generalized by record tg_1 with values vg_1, cg_1 where v and c are numerical and categorical values respectively. The distance between two records is defined as:

$$Dist(t_1, tg_1) = \sum_{i=1 \text{ to } N} Dist(c_i, cg_i) + \sum_{i=1 \text{ to } N} Dist(v_i, vg_i)$$

CHAPTER 5

SYSTEM REQUIREMENTS

All computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time.

5.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

5.1.1 Memory

All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes.

- Memory(RAM) – 4 Gigabyte

5.1.2 Processing power

The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 or x64 architecture define register size that determine the complexity of messages that can be processed, as the model and the clock speed of the CPU the requirements can range from,

- Processor – Intel / AMD
- Architecture = 32(x86) bit or 64(x64) bit
- Clock Speed = 1.5 Giga Hertz
- Number of processors = 1

5.2 Software Requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

5.2.1 Platform

A computing platform describes some sort of framework, either in hardware or software, which allows software to run. Typical Platform's include architecture, operating system, or programming languages and there runtime libraries.

- Operating System/Platform – Windows

5.2.4 Programming languages

A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

- Python is the only programming language used in the project.

CHAPTER 6
CODING

All the different code modules used in the project are presented in this chapter

6.1 Group Class Python Source Code

```
class Group(object):

    """Group is for group based k-anonymity
    self.member: record list in group
    self.gen_result: generlized value for one group
    """

    def __init__(self, member, gen_result, information_loss=0.0):
        self.information_loss = information_loss
        self.member = member
        self.gen_result = gen_result[:]
        self.center = gen_result[:]
        for i in range(QI_LEN):
            if IS_CAT[i] is False:
                self.center[i] = str(sum([float(t[i]) for t in self.member]) * 1.0 /
len(self.member))

    def display(self):
        print self.center
        print self.member

    def add_record(self, record):
        """
        add record to group
        """
        self.member.append(record)
        self.update_gen_result(record, record)
    def update_group(self):
        """update group information when member is changed
        """
        self.gen_result = group_generalization(self.member)
        for i in range(QI_LEN):
            if IS_CAT[i]:
                self.center[i] = self.gen_result[i]
            else:
                self.center[i] = str(sum([float(t[i]) for t in self.member]) * 1.0 /
len(self.member))
        self.information_loss = len(self.member) * NCP(self.gen_result)

    def update_gen_result(self, merge_gen_result, center, num=1):
        """
        update gen_result and information_loss after adding record or merging group
        :param merge_gen_result:
        :return:
        """
        self.gen_result = generalization(self.gen_result, merge_gen_result)
        current_len = len(self.member)
        for i in range(QI_LEN):
            if IS_CAT[i]:
                self.center[i] = self.gen_result[i]
            else:
                self.center[i] = str(sum([float(t[i]) for t in self.member]) * 1.0 /
len(self.member))
        self.information_loss = len(self.member) * NCP(self.gen_result)

    def __len__(self):
        """
        return number of records in group
        """
        return len(self.member)
```

6.2 Source Code Of The Algorithm

```
def my_algo(data, k=25):
    """
    Group record according to NCP
    """
    groups = []
    seed_groups = []
    small_groups = []
    # randomly choose seed and find k-1 nearest records to form group with size k
    seed_index = random.sample(range(len(data)), len(data) / k)
    for index in seed_index:
        record = data[index]
        seed_groups.append(Group([record], record))

    #remove seed records from data set
    data = [t for i, t in enumerate(data[:]) if i not in set(seed_index)]


    while len(data) > 0:
        record = data.pop()
        index = find_best_group_iloss(record, seed_groups)
        seed_groups[index].add_record(record)

    residual = []
    for group in seed_groups:
        if len(group) < k:
            small_groups.append(group)
        else:
```

```
        if len(group) > k:
            adjust_group(group, residual, k)
        groups.append(group)
    while len(residual) > 0:
        record = residual.pop()
        if len(small_groups) > 0:
            index = find_best_group_iloss(record, small_groups)
            small_groups[index].add_record(record)
            if small_groups[index] >= k:
                groups.append(small_groups.pop(index))
        else:
            index = find_best_group_iloss(record, groups)
            groups[index].add_record(record)
    print "No.of groups = ", len(groups)
    i = 0
    while i < len(groups):
        print "No.of records in group-%d= "%i, len(groups[i])
        i += 1
    return groups
```

6.3 Source Code To Calculate Distance Between Two Records

```
def r_distance(source, target):
    """
    Return distance between source (group or record)
    and target (group or record). The distance is based on
    NCP (Normalized Certainty Penalty) on relational part.
    If source or target are group, func need to multiply
    source_len (or target_len).
    """
    source_gen = source
    target_gen = target
```

```
source_len = 1
target_len = 1
# check if target is Group
if isinstance(target, Group):
    target_gen = target.gen_result
    target_len = len(target)
# check if souce is Group
if isinstance(source, Group):
    source_gen = source.gen_result
    source_len = len(source)
if source_gen == target_gen:
    return 0
gen = generalization(source_gen, target_gen)
# len should be taken into account
distance = (source_len + target_len) * NCP(gen)
return distance
```

CHAPTER 7

EXPERIMENTS AND RESULTS

The performance factors and the explanation of results are discussed in this Chapter with the respective tables, figures and screenshots

7.1 Data Set

Data Set is the collection of data which generally related to same attributes. The sample data set is like

Attribute Information:

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspect, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

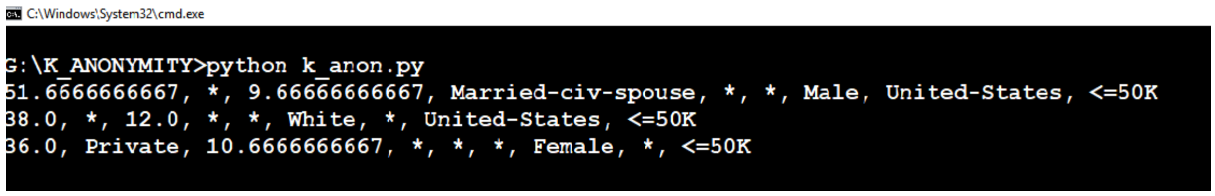
Original Data :

39, State-gov, 13, Never-married, Adm-clerical, White, Male, United-States, <=50K
50, Self-emp-not-inc, 13, Married-civ-spouse, Exec-managerial, White, Male, United-States, <=50K
38, Private, 9, Divorced,Handlers-cleaners, White, Male, United-States, <=50K
53, Private, 7, Married-civ-spouse, Handlers-cleaners, Black, Male, United-States, <=50K
28, Private, 13, Married-civ-spouse, Prof-specialty, Black, Female, Cuba, <=50K
37, Private, 14, Married-civ-spouse, Exec-managerial, White, Female, United-States, <=50K
49, Private, 5, Married-spouse-absent, Other-service, Black, Female, Jamaica, <=50K
52, Self-emp-not-inc, 9, Married-civ-spouse, Exec-managerial, White, Male, United-States, >50K
31, Private, 14, Never-married, Prof-specialty, White, Female, United-States, >50K

Data After Anonimization At Cell Level :
47.0, *, 11.6666666667, *, *, White, Male, United-States, <=50K
36.0, Private, 10.6666666667, *, *, *, Female, *, >50K
42.6666666667, Private, 10.0, *, *, *, *, United-States, <=50K

7.2 Visualization and Result

The code is executed in the command prompt and the result is shown in the images below


C:\Windows\System32\cmd.exe

```
G:\K_ANONYMITY>python k_anon.py
51.6666666667, *, 9.66666666667, Married-civ-spouse, *, *, Male, United-States, <=50K
38.0, *, 12.0, *, *, White, *, United-States, <=50K
36.0, Private, 10.6666666667, *, *, *, Female, *, <=50K
```

Fig. 7.1: Output

```
                                GROUP 1
Group's generalized value:
43.0, *, 10.3333333333, *, *, White, Male, United-States, >50K

Records in group:
52, Self-emp-not-inc, 9, Married-civ-spouse, Exec-managerial, White, Male, United-States, >50K
38, Private, 9, Divorced, Handlers-cleaners, White, Male, United-States, <=50K
39, State-gov, 13, Never-married, Adm-clerical, White, Male, United-States, <=50K

                                GROUP 2
Group's generalized value:
32.0, Private, 13.6666666667, *, *, *, Female, *, <=50K

Records in group:
28, Private, 13, Married-civ-spouse, Prof-specialty, Black, Female, Cuba, <=50K
31, Private, 14, Never-married, Prof-specialty, White, Female, United-States, >50K
37, Private, 14, Married-civ-spouse, Exec-managerial, White, Female, United-States, <=50K

                                GROUP 3
Group's generalized value:
50.6666666667, *, 8.33333333333, *, *, *, *, *, <=50K

Records in group:
49, Private, 5, Married-spouse-absent, Other-service, Black, Female, Jamaica, <=50K
53, Private, 7, Married-civ-spouse, Handlers-cleaners, Black, Male, United-States, <=50K
50, Self-emp-not-inc, 13, Married-civ-spouse, Exec-managerial, White, Male, United-States, <=50K
```

Fig. 7.2: Grouped Output visualization

CHAPTER 8

TESTING

Testing is a process which reveals errors in the program. It is the major measure employed during software development. During testing, a program is executed with a set of test cases and output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

8.1 Testing Strategies

In order to make sure that system does not have errors, the different levels of testing strategies that are applied at different phases of software development are:

8.1.1 Unit Testing

Unit testing is done on individual modules as they are completed and become executable. Each module can be tested using the following two strategies:

8.1.1.1 Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

8.1.1.2 White Box Testing

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

- Guarantee that all independent paths have been executed.
- Execute all logical decisions on their true and false Sides.
- Execute all loops at their boundaries and within their operational bounds
- Execute internal data structures to ensure their validity.

8.1.2 Integrating Testing

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

8.1.3 System Testing

It involves in-house testing of the entire system before delivering to the user. Its aim is to satisfy the user the system meets all requirements of the client's specifications.

8.2 Test Approaches

Testing can be done in two ways:

- Bottom up approach
- Top down approach

8.2.1 Bottom up Approach

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

8.2.2 Top down Approach

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred.

8.3 Test cases

- Testing is the set of activities that can be planned in advance and conducted systematically.
- The underlying motivation of program testing is to affirm software quality.

1. The k-anonymity is not valid when k value is less than two.
2. The dataset which is given to the algorithm must contain all

values.

3. Presence of incomplete and unknown values makes the anonymization very difficult hence preprocessing is required.

4. The number of clusters are defined based on the number of records and the k value.

5. The number of maximum crossovers must be comparably larger than the number of records to produce the useful crossovers.

CHAPTER 9

CONCLUSION

&

FUTURE DIRECTIONS OF WORK

The conclusions are based on the results obtained and the future work that can be done is discussed in this Chapter.

9.1 Conclusion And Future Directions of Work

This algorithm achieved a time complexity of O(n^2). This algorithm's efficiency lies in how the records in inputted original table are related to each other. The initial method used to group nearest records chooses the n/k seed records randomly. If we come up with methods that can choose the initial seed records cleverly such that the end result would be efficient then the run time of such algorithm will be increased.

CHAPTER 10
REFERENCES

1. R. Agrawal and R. Srikant, “Privacy-preserving data mining,” in Proceedings of the ACM SIGMOD International Conference on Management of Data, 2000.Y. Koren,
2. P. Samarati, “Protecting respondent’s privacy in microdata release,” *TKDE*, vol. 13, no. 6, 2001
3. L. Sweeney, “K-anonymity: a model for protecting privacy,” International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, vol. 10, no. 5,pp. 557–570, 2002
4. V. Ciriani, S. D. C. di Vimercati, S. Foresti, and P. Samarati, “K-anonymity,” Security in Decentralized Data Management (to appear).