Anonymization techniques

Semester project

AALBORG
UNIVERSITET

Semester 7 Project

Group 3

Computer Science (IT)

Aalborg Universitet

December 21st 2022

Seventh Semester - Computer Science (IT)

Selma Lagerløfs Vej 300
9220 Aalborg
http://www.aau.dk

Title:

Anonymization Techniques

Project:

Seventh Semester

Project period:

September 2022 - December 2022

Project group:

Group 3

Authors:

Iñigo Sanz
Saif Ahmed Khan
Magnus Just Sund Carlsen
Mazlum Ali Tas

Supervisor:

Hamdi Ben Hamadou

Edition: 1
Number of pages: 36
Appendix: 0
Completed: 16/12/22

The content of the report is freely available, but publication (with source reference) may only take place in agreement with the authors.

Preface

Acknowledgements

We thank Hamdi Ben Hamadou (Aalborg University) for supervising, useful discussions and providing direction throughout the course of this project.

Abstract

This report highlights the issues with sensitive data being disclosed, and aims to utilise anonymization techniques, centered around K-Anonymization and L-Diversity, to help minimise the risk of re-identification. The report delves into understanding what information a dataset could consist of and how it could potentially be at risk. Also considering existing techniques to solve the issue, and deriving an evaluation of the security in the form of metrics. The techniques consist of the previously mentioned K-Anonymization and L-Diversity, including branch off principles, such as L-Diverse, L-Entropy, and L-Recursive. The tool created, takes an arbitrary dataset and performs actions according to the techniques. As the K value increases, we receive more loss of information due to higher generalisation, less Equivalence Classes are generated, and they are larger. With a higher K value, more data is restricted. L-Diversity adds a more robust layer of protection to K-Anonymization, it ensures well represented values for sensitive data within Equivalence Classes and its distribution.

Table of Contents

Introduction 1

1.1 Introduction

In this digital age of information, data is a necessity and sensitive data is sought after by many. The average person has no idea of the value of their sensitive data. An individual's sensitive data is personal information about them, and releasing it publicly results in a breach of privacy. Unfortunately, companies are interested in such data as it describes valuable information about their customers, giving rise to insights and analyses. Moreover, this data which at first seems harmless, can lead to personal security risks such as re-identification, fraud, identity theft if this data is leaked. For instance, it is the case of the governor of Massachusetts, who got identified in the "Sweeney" medical data from 2002. [1]. In order to prevent such harmful risks, a data protection law has been established which is called GDPR [2]. This regulation allows for people to have the rights to access their own data, mainly regarding sensitive data. According to GDPR, sensitive data is considered any kind of data revealing racial or ethnic origins, health-related data including genetic and biometric, data concerning an individual's sex life or sexual orientation, as well as describing political, religious or philosophical opinions. For that reason, when an organization, such as an hospital, publishes a fragment of their patient database to support medical research, it is imperative that said sensitive data is identified and untraceable back to the individual.

To achieve this goal, instead of deleting the sensitive data -thereby removing the usefulness of the data-, a better approach is to hide an individual's quasi-identifiers. These quasi-identifiers are sets of attributes in the data that can trace back to a person, such as their age, their address or their salary. Quasi-identifiers differ greatly from explicit identifiers as the latter clearly identify individuals by storing values such as first or last names, and should be immediately removed from an anonymized dataset.

Unfortunately, anonymizing a dataset is not an easy task, as the goal is to make individuals within the dataset as similar to each other to avoid re-identification, and the input data might make anonymization difficult. This task is usually performed by considering the dataset's quasi-identifiers. A quasi-identifier is an attribute that does not identify an individual on its own, but can become identifying if paired with other quasi-identifiers. Therefore, achieving anonymization takes into consideration hiding sensitive attributes by masking any kind of trace-back to an individual with their quasi-identifiers.

However, the concept of anonymization has been researched during the past decades, and new investigations and tools show multiple possibilities to secure a dataset. In 2007, a study about anonymization using clustering tools [3] was published. It proved to be useful as clustering groups of individuals together that are similar to eachother, which is exactly the nature of dataset anonymization. Another study was published in 2007, where anonymization was utilised using spatial indexing [4]. While spatial indexing is often utilized in querying, it also proved useful in anonymization as objects are placed in a table closer to eachother based on how similar they are. Finally, in 2011, a paper was published where anonymization of social networks was achieved by vertex addition [5]. The network is built with vertex-labeled or unlabeled graphs, describing which nodes are closer -or more similar- to other nodes. As we can see, dataset anonymization usually involves studying the similarities between its individuals in order to conceal their sensitive data in an efficient manner.

This work presents an exhaustive exploration of the different anonymization techniques, i.e., K-Anonymization, L-Diverse, L-Recursive and L-Entropy, as well as multiple metrics to evaluate the generalization process, all combined into a tool that can handle both numerical and categorical data, considering multiple sensitive data.

1.2 Problem Analysis

Sensitive Data has immense value for statistics and analytics for many companies, however this data needs to be regulated. GDPR is a key aspect in user privacy. Maintaining privacy for users in Europe is essential. Therefore using software that can protect that kind of valuable data is irreplaceable. However, data is also a key aspect in consumer identification for companies across Europe, therefore it is essential to find the middle ground that both protects the user's data but also serves as a valuable tool for companies across Europe.

For example, there have been recent developments in regards to the usage of the Google Analytics tool, in late September 2022 where it was assessed by Datatilsynet in Denmark that user data is illegally sent and stored on US servers which GDPR does not regulate. This is a clear violation of the privacy metrics in GDPR and requires a new solution for many companies currently using it [6]. This issue could potentially cause further complications down the line, as they are not adhering to the regulations put in place and sensitive data is not protected [7].

How can this problem be approached?
A solution which handles sensitive data and anonymizes these before they leave European jurisdiction is therefore critical for companies to legally handle user data. While also ensuring the usefulness of said data and the value the metrics can bring. Business and organisations hold sensitive user data. The goal of that is to give the users a better experience, and also the data helps their operations become more effective. Therefore, it is essential to provide information security techniques in order to reduce malicious attacks. By using and exploring

differing techniques the solution can, based on the sensitivity of the data, alter and anonymize data in a manner to retain usefulness.

1.3 Research Question

Which anonymizing techniques can be put in place to maximize the security and protection of user data, while retaining its value and usefulness, and how would they be integrated into a tool?

1.3.1 Sub Questions

In order to accurately answer the research question, the following sub-questions will be answered through-out the paper.

What is an anonymized dataset?

How is anonymization in a dataset achieved?

What are the existing techniques to reach a certain level of anonymization?

How can these techniques and concepts be implemented into a tool?

Can certain techniques be optimized in order to ensure a better privacy, or more usefulness in the data?

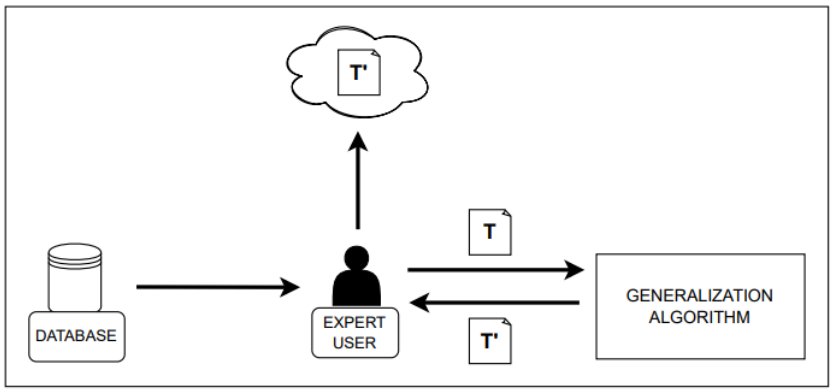
Can the anonymization of a dataset be evaluated?

System Definition 2

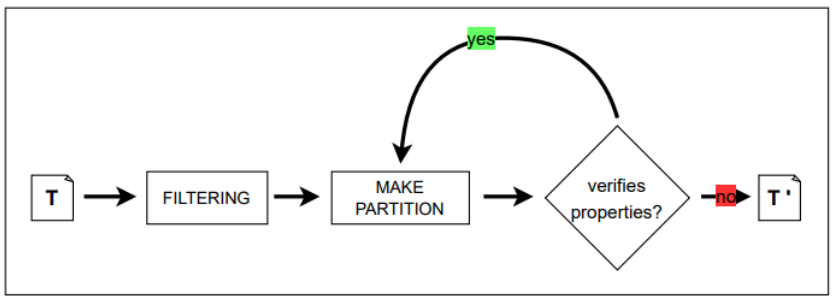
This section describes our system, and a method will be used to give a practical overview in order to evaluate the relevance of the system definition. Before the development of the system can initiate a system definition must be created in order to achieve an overview of the system characteristics that need to be fulfilled before the system can be declared done. With that being said, a preliminary analysis must be initiated, and questions such as “what problems are this system gonna solve?".

Additionally, a rich picture is also needed in order to create an overview of the system before creating the system definition. A rich picture allows the developers of the system to gain an overview of the processes the system is going through. In other words it illustrates the developer's understanding with important entities such as people, roles, and transactions that are suiting for the given environment of the respective system. [8]

2.1 Rich picture



(a): Use case of the tool



(b): Closer look to Generalization Algorithm

Figure 2.1. Diagram describing the use of an anonymization tool as well as the main idea behind it.

Figure 2.1 (a) illustrates the anonymization process that follows a secure table protecting the individuals' privacy. An existing database is used to gather all of the data that is about to become anonymized. An expert user is needed in order to identify, utilizing the GDPR regulation, the sensitive data that needs to be protected, as well as the explicit identifiers of the users. The dataset is used as input for the algorithm and an anonymized dataset in compliance with the anonymization properties is given as output. The expert user recognizes whether the anonymized data is useful for data analysis and publishes it to the web.

Meanwhile, Figure 2.1 (b) demonstrates how the algorithm that anonymizes the dataset works. The input table T undergoes a filtering process in order to remove missing values -as they cannot be generalized- as well as deleting explicit identifiers, since these give information that trace back to the individuals among the database. Afterwards, the table undergoes multiple partitions in order to place individuals with similar characteristics together. The dataset keeps being partitioned for as long as the partitioned dataset verifies the anonymization properties that the expert user needs. Once no further partitions can be made in the dataset, the output table T' is given as output.

2.2 System Definition

A computerized system is used to uphold the GDPR restrictions that are placed in countries part of the European Union. The system's goal is to anonymize user data so the individuals among the data cannot be traced back and identified. In other words, the individual data is pooled in a larger group so the information is corresponding to any user given in that specific pool. In order to anonymize the given dataset certain anonymize concepts and techniques have to be implemented in the computerized system. The data must be handled by expert users that know about GDPR and the field that the dataset is about, so that they can identify which attributes need to be protected from the public eye, and whether the resulting dataset has enough usefulness among the data [8].

2.3 FACTOR criterion

Functionality:

The system's functionality should be able to register an input dataset and produce a table with anonymized data that hides the user in plain sight. The functionality that is being applied is anonymization techniques.

Application Domain:

The application domain is controlled by organizations that work with different kinds of data. It can be the medical organization that works with patients and makes the sensitive data -their disease and biomarkers- untraceable.

Condition:

The system will be developed by students at Aalborg University, and the system will be used by users who are experienced with GDPR handling.

Technology:

The technology that will be used in this project will be the programming language Python and libraries that will help manipulate the data combined with several anonymizing techniques. Furthermore, JupyterNotebook will be used as the IDE due to the fact this project will be working with a considerable amount of data.

Objects:

The main objects of the problem domain are explicit identifiers, quasi-identifiers, and sensitive information, as well as the desired degree of anonymization.

Responsibility:

The responsibility of this system is to help companies uphold the GDPR law in order to hide their clients' sensitive information.

Materials and Methods 3

3.1 Dataset description

The dataset that we selected for the study had to be one that contained both numerical and categorical attributes. Moreover, it needed at least one sensitive attribute. For these reasons, we selected the adult dataset from the UCI Machine Learning Repository (`adni.loni.usc.edu`). This dataset contained initially 48.842 instances with 14 different attributes, however, as some individuals contained missing values, the number of individuals in the studied dataset was 45.222.

Numerical attributes

In order to understand certain anonymization decisions taken by the algorithm, it is necessary to first study the data. Numerical attributes are those that are either integers or floats, and that the distance between each other is based on how further apart they are. They form a continuous set.

Column name	**Description**	**Range**	**Mean + std.**
Age	The age of the given instance.	17-90	38.5 ± 13.2
Finalweight	Weight by Census Bureau.	13,492-1,490,400	$189,734 \pm 105,639$
Education num.	Current academic level.	1-16	10.1 ± 2.55
Capital gain	The profit the given instance earns.	0-99,999	$1,101.4 \pm 7,506.4$
Capital loss	The loss of the given instance.	0-4,356	88.6 ± 404.9
Hours per week	Hours per week of work.	1-99	40.9 ± 12.0

Table 3.1. Numerical table

For that reason, Table 3.1 describes not only what each numerical attribute represents, but also its range and distribution. As we can see, some attributes such as Capital Gain have a very considerable range, and yet their mean is extremely small, with most of the values being close to 0. Other attributes such as Age, are well distributed.

Categorical attributes

Categorical attributes are those that can be viewed as values in finite, distinct groups. These attributes do not follow the same behaviour as numerical attributes, since the distance between each other can not be measured in a continuous range. For that reason, we introduce taxonomy trees: for each categorical attribute, there is an arrangement of the values in form of a tree in which the higher the height of the node, the more generalized the value is, as opposed to the nodes being deeper into the tree, the more granular they are.

Column name	Description	Cardinality
Workclass	What sector the instances work in	8
Education	What type of education the instances have	16
Marital status	The instances marital status.	7
Occupation	The current job the instance have	14
Relationship	The relationship the instances have	6
Race	What ethnicity the instances have	5
Sex	The gender of the instances.	2
Native country	The native country of the given instance	41

Table 3.2. Categorical table

As seen in Table 3.2, some categorical attributes such as the instance's sex has a low cardinality, while the "Native Country" attribute contains a considerable amount.

Finally, we will consider the individual's ethnicity as the dataset's sensitive attribute, since it breaks GDPR compliance.

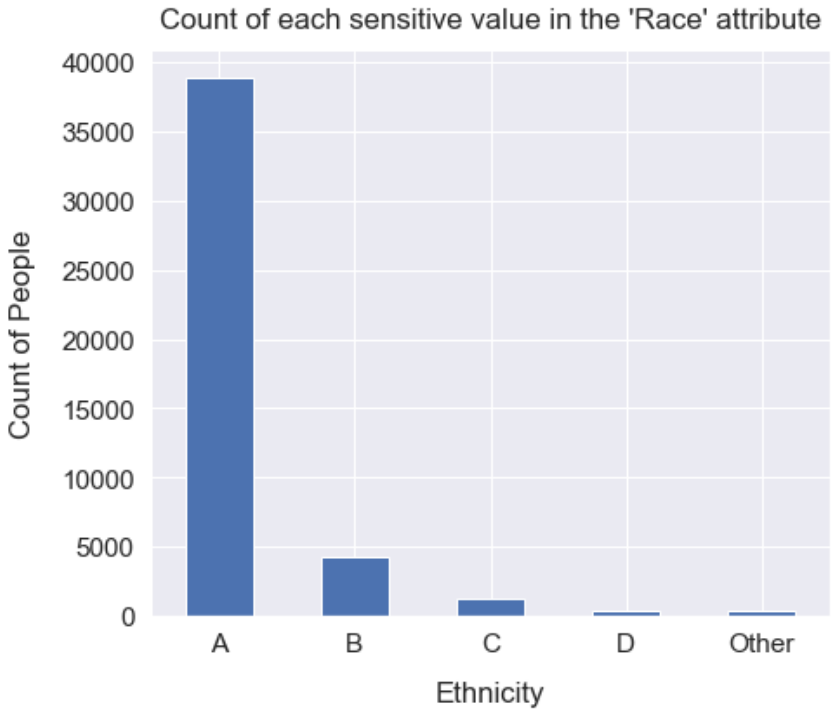


Figure 3.1. Amount of individuals in the Adult dataset of each ethnicity.

As we can observe in Figure 3.1, the distribution of the sensitive attribute is uneven. The amount of individuals of race "A" in the dataset is considerably higher than the other ethnicities. This can pose complications when anonymizing the dataset [9].

3.2 Anonymization techniques

The anonymization of a dataset can be achieved through multiple techniques that generalize the attributes of the individuals within the dataset. This generalization aims to impede adversaries from making observations and conclusions about individuals in the dataset by adding ambiguity to their data. The objective of using these techniques is to satisfy a set of properties that validate the anonymization of a dataset. An example could be initiated from this clean table in figure 3.2. figure 3.2 is a clean table that contains 6 columns with unique attributes which can be used as quasi-identifiers. There are also explicit identifiers because some attributes cannot be used as quasi-identifiers. The clean table also has a sensitive attribute column that contains sensitive information about the patients.

First name	Last name	Age	Salary	Profession	Disease
John	Roe	24	1800	Police officer	A
Maria	Garcia	35	2450	Teacher	B
Peter	Martinez	30	1900	Firefighter	B
Meredith	Malone	57	3100	Nurse	A
Hans	Pedersen	50	2750	Teacher	A
Martin	Jørgensen	62	3350	Doctor	B

Figure 3.2. Clean table

There exist multiple practices to generalize a dataset, some being more complex than others which there will be illustrated some examples later on in this section. The examples will sometimes be built upon figure 3.2 and sometimes the values in some of the attributes will be changed in order to show different anonymized table variants.

3.2.1 Suppression

Suppression is a simple technique that deletes explicit data from the dataset. As mentioned previously, explicit data is information about the user that immediately identifies them, such as an individual's first name. It can be done by either removing the attribute or removing all of the attribute's information, swapping it with an unrelated character. Keeping the attribute instead of deleting it might result in redundancy, as the table occupies more space and the data among all rows is identical. While this technique is one of the most basic ones, it is absolutely needed in order to protect the privacy of the individuals in the dataset.

First name	Last name	Age	Salary	Profession	Disease
*	*	24	1800	Police officer	A
*	*	35	2450	Teacher	B
*	*	30	1900	Firefighter	B
*	*	57	3100	Nurse	A
*	*	50	2750	Teacher	A
*	*	62	3350	Doctor	B

Figure 3.3. Example of applying Suppression on a table. The explicit identifiers were the individual's first and last name.

As we can see in Figure 3.3, no remaining information in the table can be used to immediately identify an individual in the table. However, other attributes such as their age, salary or profession can be used to track back to an individual. These are an individual's quasi-identifiers and can be handled with Generalization.

3.2.2 Generalization

Generalizing is the most common technique when anonymizing a dataset. It is a fundamental method of anonymization, as it adds ambiguity to the original data of an individual. It allows two or more individuals to have similar data, therefore complicating the adversary's goal of finding sensitive data from a specific individual. Compared to suppression, this technique does not focus on removing explicit data, but rather protecting the sensitive information of the users in the dataset. Generalization is more complex than other techniques due to its different behavior when treating numerical or categorical data. With numerical data, it replaces the individual's value with a range of values that include the original. Meanwhile, categorical data, adds ambiguity to the value. For example, if the value is a region in Denmark, we can generalize it to represent instead the country of Denmark. shifts upwards into the attribute's taxonomy tree.

First name	Last name	Age	Salary	Profession	Disease
John	Roe	20 - 30	1800	Police officer	A
Maria	Garcia	25 - 35	2450	Teacher	B
Peter	Martinez	25 - 35	1900	Firefighter	B
Meredith	Malone	55 - 65	3100	Nurse	A
Hans	Pedersen	40 - 50	2750	Teacher	A
Martin	Jørgensen	55 - 65	3350	Doctor	B

(a): Generalization applied to the Age column in sets of 10

First name	Last name	Age	Salary	Profession	Disease
John	Roe	24 - 30	1800	Police officer	A
Maria	Garcia	35 - 50	2450	Teacher	B
Peter	Martinez	24 - 30	1900	Firefighter	B
Meredith	Malone	57 - 62	3100	Nurse	A
Hans	Pedersen	35 - 50	2750	Teacher	A
Martin	Jørgensen	57 - 62	3350	Doctor	B

(b): Generalization applied in regards to the individual values

Figure 3.4. Generalization

In Figure 3.4 (a), the specific values in the age column have been modified with ranges containing their original value. With this method, significant information regarding this column becomes more ambiguous. Note that while in this figure the values range in groups of ten, the data becomes less useful since there is added noise. For that reason, generalization based on values between individuals makes the generalized table more useful, as we can see in Figure 3.4 (b). By using this approach, the goal of the anonymized table is to keep generalizing the attributes until the individuals cannot be traced back.

It is important to note that generalization has been chosen over swapping values of the same attribute because the latter will make the data less useful. For example, if we have two individuals and two or more attributes, swapping one individual's attribute with the other will lead to erroneous conclusions concerning multi-dimensional analyses.

3.3 Anonymization concepts

This section will illustrate and explain the different anonymization concepts which have been implemented in the system. For the sake of demonstrating the examples clearly, the attributes will have different values assigned. The following concepts in this section will have illustrations in order to show the implementation and an explanation of their capabilities. Furthermore, some vulnerabilities will also be explained in order to show that even if some of the concepts are implemented the data is still not safe from adversaries.

3.3.1 K-Anonymization

K-Anonymization is the fundamental property to anonymize a dataset. For a dataset to satisfy this property, it is necessary that any person's quasi-identifiers within the dataset are indistinguishable from at least $k - 1$ other individuals. In order to make an individual indistinguishable from another, it is necessary to make use of the techniques described in the section above, mainly through generalization. With this technique, the security and privacy of the individuals in the dataset increase, where each individual has a probability of $\frac{1}{K}$ of being recognized, and the usefulness of the data lowers, since most of it has been generalized.

First Name	Last Name	Age	Salary	Profession	Disease
*	*	25 - 30	1800 - 2300	Public safety	A
*	*	25 - 30	1800 - 2300	Public safety	B
*	*	25 - 30	1800 - 2300	Public safety	B
*	*	31 - 35	2450 - 2800	Health service	A
*	*	31 - 35	2450 - 2800	Health service	A
*	*	31 - 35	2450 - 2800	Health service	B

Figure 3.5. Table satisfying K-Anonymization for $K = 3$

In figure 3.5 there are 6 columns with their own respective attributes. In other words, it references the equivalence class in values. In our example, there are 6 columns. The column marked in red is the sensitive attribute that describes the disease the patient has. The table is generalizing quasi-identifiers. The name and last name is suppressed because they are explicit identifiers. However, the column age, salary, and profession can be generalized in 3 equivalence classes because it is K-3. Therefore the age, salary, and profession should have 3 rows between 25 - 30, and the salary between 1800 - 2300, and their profession which is public safety. Each of these values is now generalized in K3 where the values can be anything specific. Such as row 1 could have the age 27, and salary of 1.900, and be a nurse. That is just an example, and something a user with restricted access would not know.
K-Anonymization does not seem to be enough to protect the individuals within the dataset. When this property is satisfied, the individuals' quasi-identifiers are indistinguishable from one another however, their sensitive data has been kept untouched, giving rise to two possible vulnerabilities.

Homogeneity Attack

The first vulnerability is the Homogeneity Attack, which can take place within a set of K records. If the sensitive values for this set of records are identical then we know that any individual in the dataset with those quasi-identifiers has that sensitive value. In other words, if the specific patient from Figure 3.5 has disease A and is between the age group 31 - 35 then it can be traced back to the specific person based on their homogeneity. It is also demonstrated in figure 3.6 below.

Background knowledge Attack

The second vulnerability is the Background Knowledge Attack. If the sensitive values within the set are not identical but the adversary knows information about the victim to rule out certain values, the adversary can figure out private information about the victim. An example can again be the patient between the age group 31 - 35 from Figure 3.6. Based on the background knowledge the adversary has then the patient with disease A can be traced given that the adversary has the required background knowledge.

First name	Last name	Age	Salary	Profession	Disease
*	*	25 - 30	1800	Police officer	A
*	*	25 - 30	2450	Teacher	B
*	*	31 - 35	1900	Firefighter	B
*	*	31 - 35	3100	Nurse	A
*	*	36 - 40	2750	Teacher	A
*	*	36 - 40	3350	Doctor	B

Figure 3.6. Homogeneity Attack and Background Knowledge Attack

Figure 3.6 which is above can be either a homogeneity attack or a background knowledge attack. For this example, only the background knowledge attack will be described. A use case scenario would be that an uninvited guest has somehow gotten access to the patient's journal because they want to know the disease of a certain person. Even if the identity of the patient is anonymized, the adversary can use their knowledge to track down the specific patient. If the adversary knows the salary of the person and which age group they are in, then they can easily make an exclusion method to remove the unwanted data and therefore track down the desired patient. These vulnerabilities pose a great threat to the security of the individuals in the dataset, giving rise to a more advanced concept, named L-Diversity, that builds on top of K-Anonymization.

3.3.2 L-Diversity

L-Diversity is required in a public dataset because, while K-Anonymization focuses on its quasi-identifiers, L-Diversity focuses on the sensitive values of the individuals. Just like with K-Anonymization, this concept poses a trade-off between the usefulness of the data and the privacy and security of the individuals in the dataset.

The principle of L-Diversity is to have at least L well-represented values for the sensitive attribute in each equivalence class, i.e., for every q^*-group of a published table T^*. There are three types

of L-Diversity that guarantee that the sensitive values within an equivalence class are well-represented. [10]

First Name	Last Name	Age	Salary	Profession	Disease
*	*	25 - 30	1800 - 2450	Public safety	A
*	*	25 - 30	1800 - 2450	Public safety	A
*	*	25 - 30	1800 - 2450	Public safety	B
*	*	36 - 40	2750 - 3350	Health service	A
*	*	36 - 40	2750 - 3350	Health service	B
*	*	36 - 40	2750 - 3350	Health service	B

Q1

Q2

Figure 3.7. L(2) - diversity table

In Figure 3.7 above, which is L-diversity they have been put into 2 groups. L-Diversity is used in order to prevent homogeneity attack and knowledge background attack. It is an over extension of K-Anonymization. All the quasi-identifiers from K-Anonymization must be in the same group in L-Diversity as can be seen in Q1 and Q2. There are 2 different diseases in this table, since the L-Diversity is assigned to the value 2 then the record should have 2 different values in each group.

L-Diverse

An L-Diverse dataset is a technique where the set of sensitive values in each equivalence class contains at least L distinct sensitive values. This concept greatly increases the security of the individuals within a dataset due to invalidating attacks such as the Homogeneity Attack mentioned previously as one of the flaws of K-Anonymization. The higher the degree of L, the more secure the dataset will be.

Though L-Diversity does not prevent attacks from adversaries with arbitrary background knowledge.

$$\exists s, \quad \frac{f(s'|q)}{f(s'|q*)} \approx 0 \tag{3.1}$$

Since it is not possible to protect against arbitrary amount of background knowledge an adversary might have, above equation states a problem with the L-Diversity approach. Above equation states that an individual with a quasi-identifier q is much less likely to have a sensitive attribute s' if an adversary with strong background knowledge can be sure that the difference of prior and posterior knowledge is 0. This is where different approaches is considered to strengthen

the L-Diversity implementation. When instantiating L-Diversity there are different approaches which is L-Entropy and L-Recursive.

Entropy L-Diversity

As Distinct L-diversity allows for a custom field, to determine the amount of distinct values required for the sensitive field. Entropy L-Diversity requires a distribution of sensitive values in each equivalence class to be at least equal to $log(l)$. L-Entropy ensures that an adversary can not use the cardinality of sensitive attributes as an affirmative prediction of someone having a specific disease. Let there exist 2 unique sensitive attributes in a given data set of size 500. Let one of the two unique sensitive attributes occurs 400 times then the probability of guessing a sensitive value for a given person is 80 percent. That is without taking into account the quasi-identifiers. If it is now known that a sensitive attribute occurs infrequently, per chance under 10 times then the probability of a person connected to this attribute is 2 percent [10]. Entropy L-Diversity makes sure that the probability for an adversary guessing a sensitive value of an individual using the probability of occurring sensitive value in a dataset is minimized. A formula for checking that an EQ is satisfying the Entropy L-Diversity, the following is used:

$$-\sum_{s \in S} P(qid, s) * log(P(qid, s)) \geq log(l) \tag{3.2}$$

It says that S is the set of sensitive values and where each sensitive value s in each EQ, the probability is calculated for each s appearing. The end result is negated and compared to the $log(l)$. If the formula is satisfied, then the EQ satisfy Entropy L-Diversity. For a dataset to satisfy Entropy L-Diversity all EQs have to satisfy the above formula. Though this leads to some limitations which is mentioned in [Amin Aminifar, 2011][10]. The Entropy L-Diversity is quite strict and can cause datasets with a lot of the same sensitive value, to require a high level of L-Diversity for well represented sensitive values, which can cause a larger loss of information. This is where the Recursive L-Diversity can ensure that a sensitive attribute does not occur either too frequently or infrequently.

Recursive L-Diversity

If every quasi identifier block satisfy Recursive L-Diversity block then the anonymized table satisfies the Recursive L-Diversity. The formula for Recursive L-Diversity we have the following variables: Let S be a set of all senstive values in the dataset and r_i where i^{th} is the most frequent sensitive value that appears in the quasi identifier block and r is the denoted value and sorted in decreasing order. Then each sensitive value is represented as $n_1, n_2, ..., n_m$ where each $n \in S$. Then a constant c such that $c > 0$, that is user-defined, is then used for each EQ group such that:

$$r_1 \leq c \sum_{i=l}^{m} r_i \tag{3.3}$$

The most frequent value in an EQ group is compared to the sum, r_i, of every other sensitive value which is multiplied with the constant c which checks the frequency of each sensitive value across each EQ group. If each EQ group satisfy above formula, then the dataset is said to be Recursive L-Diversity. By this we can increase or decrease the c value such that it meets the L-Diversity, which is more flexible compared to Entropy L-Diversity [10].

3.4 Measurement metrics

As mentioned in chapter 3, there is a large pool of available anonymization techniques, however, their use is not straight-forward, and anonymization can be handled in different ways. For that reason, in this section we will introduce a number of metrics that can be used to evaluate how the anonymization process was performed [11]. These metrics must respond to the sub-questions proposed in section 1.3.1, since they must give information regarding how useful and secure the data in the generalized table is, as well as information regarding the partitions created for generalization.

3.4.1 Discernibility Metric

The Discernibility Metric describes how distinct a record is from others. Values in K-Anonymization models are altered either by suppression or by generalization in operations. These operations create groups of records that consist of the same values. They are called Equivalence classes (EQs). The Discernibility Metric assigns a penalty to each record that which are equal to the size of the equivalence class. This metrics' purpose is to illustrate EQs contain more information loss and therefore show that lower values are more desirable [11].

$$DM(T^*) = \sum_{\forall EQs.t.|EQ| \geq k} |EQ|^2 + \sum_{\forall EQs.t.|EQ| < k} |T| * |EQ| \tag{3.4}$$

In $DM(T*)$, T stands for the original table before being anonymized. $|T|$ is the number of records and $|EQ|$ is the size equivalence classes created in anonymized groups which means after the anonymized table have been created. The formula can be understood as for equivalence classes that is larger or equal than K where the summation is EQ to the power 2 is being added with summation of the number of records multiplied with the size created in anonymized group where every EQ is smaller than K [11].

3.4.2 Average Equivalence Class Size Metric

The average equivalence class size metric (CAVG) measures the degree of effectiveness of the creation of equivalence class (EQ) approaches the best case such that each record is generalized in an EQ of K-records. [11]

$$C_{AVG}(T^*) = \frac{|T|}{|EQs| * k} \tag{3.5}$$

CAVG is measured for the anonymized table is to first declare the original table, T. The number of records in $|T|$ needs to be divided with the number of $|EQ|$ multiplied with the value of K which is the privacy requirement. [11]

3.4.3 Information Loss

Information Loss metrics measures the values lost in a operation. The information loss is calculated in two different formula which depends on their category which is either numerical or categorical. [12]

Numerical formula

This formula calculates the normalized certainty penalty for attributes that have numerical values. The numerator and denominator represent the range of the attribute (A) for the given class which is G and the entire table. [12]

$$NCP_{A_Num}(G) = \frac{max^G_{A_{Num}} - min^G_{A_{Num}}}{max_{A_{Num}} - min_{A_{Num}}} \tag{3.6}$$

Categorical formula

The categorical formula is different from the numerical formula because there is no order and distance function. In all A_{Cat} included in G, u is the common lowest ancestor. The number of leaves is defined by $card(u)$ and is then divided with A_{Cat} which represent each unique value of A_{Cat} values. [12]

$$NCP_{A_{Cat}}(G) = \begin{cases} 0, & \text{card(u)} = 1 \\ card(u)/|A_{Cat},| & \text{otherwise.} \end{cases} \tag{3.7}$$

The NCP of class G's attribute can be defined with the following formula:

$$NCP(G) = \sum_{i=1}^{d} w_i * NCP_{A_i}(G) \tag{3.8}$$

D holds the value for the amount of attributes in the dimension. A_i can either be categorical value or numerical value and has the weight w_i where the summation of w_i is 1. The information loss is only measured for a single equivalence class in NCP, which is the reason GCP is used in order to measure the information loss for the entire table. [12]

$$GCP(\mathcal{P}) = \frac{\sum\limits_{G \in \mathcal{P}} |G| * NCP(G)}{d * N} \tag{3.9}$$

In the GCP formula P is the power set of all equivalence classes in the anonymized table. N holds the value of number of records in the original table, and d stands for the dimensionality. GCP holds values between 0 and 1 where 0 signifies no information loss and 1 signifies a total information loss. [12]

Implementation 4

4.1 Partitioning a dataset

The manner in which partitioning is handled is similar to the Mondrian algorithm proposed by LeFevre, which ensures a K-Anonymization degree for a numerical dataset [13]. The Mondrian algorithm begins the process of anonymization by generalizing the dataset entirely, meaning that there exists only one equivalence class at the beginning. Moreover, each attribute is represented by one dimension. First and foremost, a dimension must be picked based on a heuristic, and LeFevre proposes to pick the dimension with the largest normalized width. Afterwards, if the anonymization properties are still upheld, the chosen dimension is split by the median, meaning that all values equal or lower than the median are placed into a new dimension, and the rest of values placed into another dimension. We keep repeating this process until no more dimensions can be split without breaking K-Anonymization. It is important to mention that for each row in the dataframe, a new column named "partition" has been created which describes the Equivalence Class for a given row. The value of "partition" is a concatenation of the values of the quasi-identifiers. With this new column we can easily gather the rows that are part of the same Equivalence Class.

4.1.1 Dimension Class

As proposed by LeFevre's Mondrian algorithm [13], the process of anonymizing a dataset begins with generalizing it as much as possible, where each attribute is described by only one dimension. Subsequently, new dimensions will be created off the original ones depending on whether the anonymization properties are upheld. Note that the Mondrian algorithm is limited to numerical values, therefore we expanded it to categorical values to create partitions containing both numerical and categorical data.

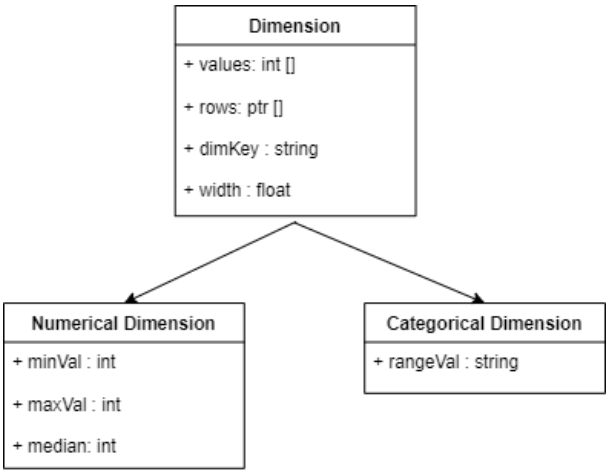


Figure 4.1. Definition of a Dimension.

We represent partitions on a dataset with the Dimension class, by representing the separations within each attribute in the dataset. Since numerical and categorical attributes have different mannerisms when considering anonymization, a NumericalDimension class was made to treat numerical dimensions, and a CategoricalDimension class was made to handle categorical ones. This class is useful to check which dimensions can be split and which cannot. As the number of rows within a dimension gets smaller, so do the chances of satisfying K-Anonymity and L-Diversity for the newly-created dimensions.

As seen in Figure 4.1, we have a class Dimension which contains the values of the attributes, as well as the individuals that are part of it. Moreover, a string is needed in order to know which attribute this dimension describes. Finally, each dimension has an attribute called width which dictates the priority of the dimension being selected to be split. Its inherited classes, NumericalDimension and CategoricalDimension contain the remaining attributes to fully describe each type of dimension.

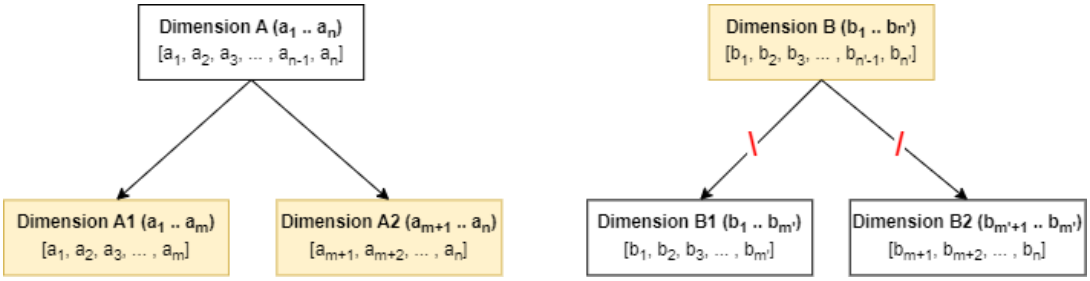


Figure 4.2. Two numerical dimensions A and B are split by their median value. The partition on A is successful while the partition on B breaks the anonymization properties.

The NumericalDimension class contains a minimum and maximum value to describe the width of the values it holds, as well as the median in order to know the splitting point for this dimension. For numerical dimensions, its width is calculated as LeFevre proposed: $(maxV - minV)/maxV$,

with maxV being the maximum value in the dimension and minV its counterpart.

As seen in Figure 4.2, a numerical dimension A is split through a value picked by a heuristic, in this case the median. Since it upholds the anonymization property, two new dimensions are created separating the initial values. Afterwards, splitting the numerical dimension B doesn't satisfy the anonymization properties anymore as dimension A has already created a partition within its rows. Therefore, dimension B won't be allowed to split again. At the end of the process, two equivalence classes have been obtained: $A1, B$ and $A2, B$. This example denotes the importance of the order in which the dimensions are split.

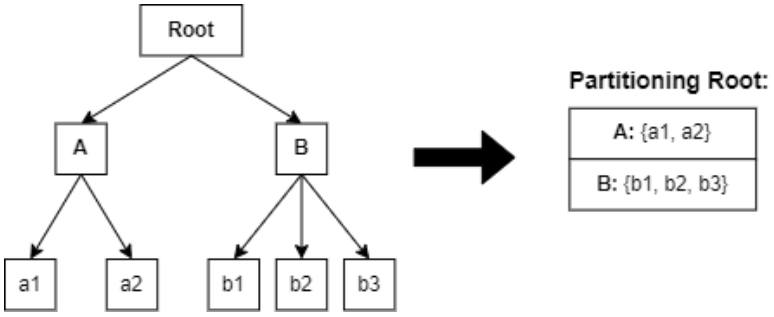


Figure 4.3. Example of a taxonomy tree. Note that by partitioning Root, we create two new dimensions A and B. As children of A; $a1$ and $a2$ will be part of that dimension, as opposed to B and its children $b1$, $b2$ and $b3$.

Regarding the Categorical Dimension class, it contains a variable that specifies the value that it describes in the taxonomy tree. The width for these dimensions can be calculated as: $1 - \frac{1}{|set(V)|}$ with $|set(V)|$ representing the cardinality of unique values in the dimension. It is important to note that the values described by the dimension are all of those which are children of the value that the dimension currently represents. Figure 4.3 describes this idea, if we split from the root, we end up with two dimensions A and B, the former only containing values that are children of A and the latter containing only values that are children of B.

It is important to note that complementary weights can be assigned to the attributes in order to give more or less priority to the order in which they are split, such that $newWidth = width \times weight$, as the partition order affects the end result.

4.1.2 K-Anonymization

As explained in the section above, the main idea of the algorithm is to begin with a generalized dataset and gradually remove the generalization among its attributes.

The core section of the algorithm can be seen in Algorithm 1. We begin the process by creating a dimension for each attribute depending on whether they are of numerical or categorical nature as seen in lines 1 to 7. D represents a list of dimensions, where $canSplit(D[i])$ dictates whether the dimension D_i can be split or not. The while loop in line 8 ensures that for as long as

Algorithm 1 Anonymize dataset X

Input X: DataFrame; D: list of dimensions; A: list of attributes; x_K: wanted degree of K; x_L: wanted degree of L

Output Anonymized dataset

1: **for** $i < len(A)$ **do**
2: **if** $A[i]$ is categorical **then**
3: $D[i] \leftarrow$ CategoricalDimension($X[i]$) {*we generate dimensions for each of the attributes*}
4: **else**
5: $D[i] \leftarrow$ NumericalDimension($X[i]$)
6: **end if**
7: **end for**
8: **while** $canSplit(\exists D) = true$ **do**
9: $d \leftarrow$ choose_dimension(D) {*if there is a splittable dimension, we select one of them based on which has the largest width.*}
10: **if** d is numerical **then**
11: $s, lhs, rhs \leftarrow$ split_dimension_numerical($D[d]$, x_K, x_L)
12: **if** $s == true$ **then**
13: update(D, lhs, rhs) {*if the partition is valid, we update the list of dimensions with the new ones*}
14: **else**
15: d.splittable = $false$ {*if the partition is not valid, the chosen dimension can no longer split again.*}
16: **end if**
17: **else**
18: $s, dims[] \leftarrow$ split_dimension_categorical($D[d]$, x_K, x_L)
19: **if** $s == true$ **then**
20: update(D, $dims[]$)
21: **else**
22: d.splittable = $false$
23: **end if**
24: **end if**
25: **end while**
26: **return** X

there is a splittable dimension in D, we choose a splittable dimension and we make a new partition based on whether it is numerical or categorical, as seen in lines 10 and 17. If the new partition upholds the anonymization properties, we then add the new dimensions in the list and remove the original one. On the other hand, if the new partition is not suitable, then the chosen dimension can no longer opt to split again. Note that creating a new partition off a numerical dimension returns two new dimensions lhs and rhs because of the separation through the median, whereas the categorical dimension returns as many new dimensions as there are children in the taxonomy tree. The computational cost of this algorithm in the worst case is $O(|A| \times |X| + |D| \times f_N \times f_C)$ with f_N denoting the computational cost to split a numerical dimension, and f_C the computational cost to split a categorical dimension.

Splitting numerical dimensions

For K-Anonymization purposes, LeFevre's proposal of partitioning a numerical dimension by the median works well in most cases, however there are some situations in which exceptions should be made. First and foremost, if the range of a dimension is high but its distribution is uneven, with most values being very similar to each other, its partitioning will return a dimension with an extremely small range and another one with a very considerable range. In the case of the adult dataset, the attribute "capital gain" ranges from 0 to 99,999. However the average and standard deviation of said attribute is 1101.43 ± 7506.43 and its median is 0. This entails that, with the Mondrian algorithm, one of the dimensions we would obtain describes all the smaller values than the median which are all equal to 0 and therefore does not add ambiguity. For that reason, this tool considers multiple candidates to split a dimension by other than the median, for example quarterlies $Q1$ and $Q3$. The partition will be made around the candidate that returns a satisfactory K value.

Moreover, it is important to also consider satisfying the properties regarding L-Diversity. Depending on what type of L-Diversity is at hand, partitioning a dimension by the median will not always return a satisfactory L value, therefore considering multiple candidates. It is important to find the best possible L value as two partitions that have the same K value can differ regarding to L-Diversity.

Algorithm 2 describes the process to a split a numerical dimension. This process consists in first generating the possible candidates to split the dimension by, as seen in line 1. For each candidate (line 3), we generate two dummy dimensions resulting of splitting the original dimension by that candidate, as described in the for lop in lines 5 to 11. If the new partition satisfies K-Anonymity and L-Diversity, and no other candidate obtains a better L value, then the partition is confirmed, in lines 17 and 18. If no candidate was able to uphold the conditions, then the function returns false which will in turn make the original dimension no longer considered for further partitions.

Splitting categorical dimensions

Effectively splitting categorical dimensions can be performed by the use of taxonomy trees. The tool considers such trees in .txt files that describe the hierarchy through the format: $parent : child1, child2, \ldots, childN$. One of the parents has to be named "Root", denoting the top of the taxonomy tree. Each categorical attribute needs to be represented with a taxonomy tree.

Algorithm 3 explains the splitting process. In line 1, we first check whether the node represented by the dimension is included in the taxonomy tree, but is not a leaf. If it is not a leaf, it means that the value can become more specific, creating a new dimension for each child of the node. This functionality is done in the for loop in line 4, where *node* represents each child of the node and we check whether each categorical value of the dimension D can be described by *node*. This

Algorithm 2 Splitting a numerical dimension D

Input D: Dimension

Output Boolean describing valid partition and two new dimensions

1: $C \leftarrow$ candidate_function(D) {*we obtain candidates based on the candidate function to check the anonymization properties*}
2: $bestDims = []$
3: **for** $c < |C|$ **do**
4: $lhs, rhs = []$ {*we check for each candidate whether the resulting dimensions satisfy anonymization*}
5: **for** $i : \forall D.values$ **do**
6: **if** $D.values[i] < c$ **then**
7: append$(lhs, D.v[i])$
8: **else**
9: append$(rhs, D.v[i])$
10: **end if**
11: **end for**
12: $lhsD \leftarrow$ NumericalDimension(lhs)
13: $rhsD \leftarrow$ NumericalDimension(rhs)
14: $s_K \leftarrow$ satisfiesKAnonymity()
15: $s_L, deg_L \leftarrow$ satisfiesLDiversity()
16: **if** $s_K \wedge s_L \wedge deg_L > max_L$ **then**
17: $max_L = deg_L$ {*if the properties are upheld and we obtain a better L degree, we keep these new dimensions*}
18: $bestDims = [lhsD, rhsD]$
19: **end if**
20: **if** $bestDims \neq []$ **then**
21: return $true, bestDims[0], bestDims[1]$ {*if the split is valid, we return true along with the new dimensions*}
22: **else**
23: **return** $false, null, null$
24: **end if**
25: **end for**

check is performed by the auxiliary function is_parent_of, in line 7, which lets us know if $D.values[i]$ is a subset of *node*.

Verifying degree of K

For each new partition, we have to verify whether the anonymization properties have been satisfied.

Algorithm 4 verifies these properties for a given degree of K. In the for loop in line 2, we count the amount of individuals within each existing Equivalence Class generated by the previous partitions. If any count is less than the desired degree of K, then the properties for K-Anonymization have not been upheld.

Algorithm 3 Splitting a categorical dimension D

Input D: Dimension; T: taxonomy tree for the categorical attribute
Output Anonymized dataset

if $D.rangeVal \in T \wedge D.rangeVal \neq$ Leaf **then**
  $N = T[D.rangeVal]$ {*if the value represented by D is in the taxonomy tree and is not a leaf, we gather its children*}
  dims = []
  for $node : \forall N$ **do**
    newV = [] {*we aim to create a new dimension for each child*}
    for $i : \forall D.values$ **do**
      if is_parent_of($node, D.values[i]$ **then**
        append(newV, $D.values[i]$) {*the value represented by the original dimension can be moved down to a more specific one*}
      end if
    end for
    newD = CategoricalDimension(newV) {*a dimension is created with the new values*}
    $s_K \leftarrow$ satisfiesKAnonymity()
    $s_L, deg_L \leftarrow$ satisfiesLDiversity()
    if $s_K \wedge s_L$ **then**
      append(dims, newD) {*if the anonymization properties are satisfied, then we add the new dimension to the dimension list*}
    end if
  end for
  return dims
else
  return *null*
end if

Algorithm 4 K-Anonymization verification

Input X: DataFrame; K: wanted degree of K
Output Bool describing successful partition

P = X['EQ'] {*we find the different equivalence classes in the dataset.*}
for p ∈ P **do**
  if count(X, p) $< K$ **then**
    return False {*if the number of individuals in the dataset following partition p is lower than K, the property is not satisfied.*}
  else
    return True
  end if
end for

4.2 Analyzing sensitive values

As mentioned through-out the report, there exist multiple options when checking for L-Diversity in the anonymized dataset. A common template was made for these different concepts as they all work in a similar manner.

Algorithm 5 Main template for L-Diversity verification

Input X: DataFrame; S: sensitive attributes; L: required L
Output *Bool*, degree of L

1: **if** $S == []$ **then**
2: **return** $True$, 0 {*if there are no sensitive attributes, no need to check for L-Diversity.*}
3: **else**
4: initial$_L$ = []
5: **for** $s \in S$ **do**
6: append(initial$_L$, $|set(X[s])$ {*we gather how many distinct sensitive values are in each attribute s.*}
7: **end for**
8: global$_L$ = min(initial$_L$) {*the sensitive attribute with less distinct values is selected as a lower bound.*}
9: **for** $s \in S$ **do**
10: $P = X$['EQ'] {*we get all the equivalence classes already made.*}
11: score$_L$ = LVerification(X, s, P) {*the nature of the function depends on whether we are using LDiverse, LEntropy or LRecursive.*}
12: **if** score$_L$ < global$_L$ **then**
13: global$_L$ = score$_L$ {*if the new L value for a given s is lower than the previous L value, we update it.*}
14: **end if**
15: **end for**
16: **if** global$_L \geq L$ **then**
17: **return** $True$, global$_L$ {*if L-diversity has been verified, we return True and the degree of L.*}
18: **else**
19: **return** $False$, global$_L$
20: **end if**
21: **end if**

Algorithm 5 illustrates the template for L-Diversity verification. First and foremost, in line 1 we check if there are no sensitive attributes in S. If no sensitive attributes can be evaluated, then L-Diversity does not exist in the dataset. If there are, we need to check for each sensitive attribute its distinct values, as done in line 6. The starting degree of L will be the smallest number of distinct sensitive values in one of the attributes. Note that when the diversity properties cannot be upheld for L, they can still be satisfied by values lower than L.

Afterwards, in line 11, the function **LVerification** represents either L-Diverse, L-Entropy or L-Recursive, as the user decides. This function takes as input the dataframe, as well as the sensitive attribute it needs to consider and the different Equivalence Classes. It will return the degree of L of the dataframe, which will be checked in line 12 whether it is lower than the degree

of diversity we calculated previously. The verification is done for each of the sensitive attributes and the verification with the smallest degree of L is kept. Moreover, if the minimal degree of L obtained through each iteration is greater or equal than the desired degree of L, then the diversity properties have been satisfied.

4.2.1 L-Diverse

In order to check that the L-Diverse property is upheld, we must check for each existing Equivalence Class how many sensitive values it contains.

Algorithm 6 L-Diverse verification

Input D: DataFrame; s: sensitive attribute; P: list of EQs
Output new L degree

1: $\text{score}_L = |set(D[s])|$ {*set of distinct sensitive values for the whole dataset.*}
2: **for** $p \in P$ **do**
3: $\text{local}_L = |set(D[s])|$ where $D[\text{'EQ'}] == p$ {$local_L$ *checks how many distinct sensitive values a given EQ p contains.*}
4: **if** $\text{local}_L < \text{score}_L$ **then**
5: $\text{score}_L = \text{local}_L$
6: **end if**
7: **end for**
8: **return** score_L

In Algorithm 6, we find the implementation of L-Entropy. In line 1 we obtain the set of distinct sensitive values in the dataset. This is the upper-bound for any possible L degree that we can find among the dataset, let us call it $score_L$. Subsequently, in the for loop we check that each EQ has at least $score_L$ sensitive values. If the check is successful, then we return the degree of L, and the next sensitive attribute in Algorithm 5 will be checked. If it is not successful, we return the lower degree of L which will be considered as the new degree of L for the whole dataset. The computational cost of this algorithm is $O(|P|)$, with P being the amount of Equivalence Classes in the dataframe.

4.2.2 L-Entropy

For the L-Entropy property to be satisfied, the entropy of each Equivalence Class -or set of quasi-identifiers-, has to be lower than $log(L)$.

In Algorithm 7, we find the implementation of L-Entropy. The aim of this algorithm is to compute the entropy for each Equivalence Class and compare it with the condition to satisfy L-Entropy. In line 6 we obtain a smaller part of the DataFrame to gather all rows that are part of the Equivalence Class that is going to be analyzed. In line 8 we obtain the different frequencies by looking at the distinct sensitive values among the Equivalence Class. Then in lines 11 to 13 we obtain the entropies for each one of the sensitive values, followed by the computation of the entropy for the whole Equivalence Class. We then check in line 15 when the entropy of

Algorithm 7 L-Entropy verification

Input X: DataFrame; s: sensitive attribute; P: list of EQs
Output score_L

1: $\text{score}_L = |set(X[s])|$
2: **for** $p \in P$ **do**
3: $\text{local}_L = \text{score}_L$
4: frequencies = [] {*array for the frequency of each sensitive value in s*}
5: entropies = [] {*array to calculate entropy of each EQ*}
6: X_S = X where X['EQ'] == p {*we obtain the rows in DataFrame X for each existing EQ*}
7: uniqueValues = $|set(X_S[s])|$ {*we compute the amount of sensitive values in the reduced DataFrame*}
8: **for** $i \in$ uniqueValues **do**
9: append(frequencies, $count(X_S.values, i)/|X.values|$) {*the frequencies are computed for each sensitive value in the EQ.*}
10: **end for**
11: **for** $i \in$ frequencies **do**
12: append(entropies, $i \times \log(i)$) {*the entropy is calculated multiplying by the logarithm of the frequency.*}
13: **end for**
14: EntropyQI = -($\sum$(entropies)) {*the entropy for the EQ is computed and compared with the condition to satisfy L-Entropy*}
15: **while** EntropyQI $< \log(\text{local}_L) \wedge \text{local}_L > 0$ **do**
16: local_L -= 1
17: **end while**
18: **if** $\text{local}_L < \text{score}_L$ **then**
19: $\text{score}_L = \text{local}_L$
20: **end if**
21: **end for**
22: **return** score_L

the Equivalence Class is smaller than the logarithm of L. If it is, we return the degree of L, if not, we reduce L by 1 and keep checking, until L is 0. The computational cost for L-Entropy is $O(|P| \times |s|^2$ with P being the number of Equivalence Classes and $|s|$ the length of the set of sensitive values in the sensitive attribute.

4.2.3 L-Recursive

To satisfy the L-Recursive property, we have to count the number of times each sensitive value appears in each Equivalence Class, and compare it with the sensitive value that is most frequent.

In Algorithm 8, we find the implementation of L-Recursive. In line 2 we count for each Equivalence Class the amount of times that a sensitive value appears in X. We then sort the list in descending orden in line 10, so that we can calculate the criteria that satisfies this property, in line 11. The while loop in line 12 checks that the most frequent sensitive value appears less times than the rest of sensitive values starting from $L - 1$. If it appears less times, then the property is satisfied. If not, the degree of L is decreased and the satisfaction criteria

Algorithm 8 L-Recursive verification

Input X: DataFrame; s: sensitive attribute; P: list of EQs; c: recursive factor
Output new L degree

1: $\text{score}_L = |set(X[s])|$ {*set of distinct sensitive values for the whole dataset.*}
2: **for** $p \in P$ **do**
3: $\text{local}_L = \text{score}_L$
4: counts = []
5: X_S = X where X['partition'] == p
6: uniqueValues = $|set(X_S[s])|$
7: **for** $i \in$ uniqueValues **do**
8: append(counts, count(X.values, i)
9: **end for**
10: sort(counts) {*returns list in descending order.*}
11: satCriteria = $\sum$counts[$\text{local}_L - 1$: |counts|] $\times$ c {*elements in list count from position* $local_L - 1$ *to the end are summed up and multiplied by the recursive factor.*}
12: **while** counts[0] >= satCriteria $\wedge$ $\text{local}_L > 0$ **do**
13: local_L -= 1
14: satCriteria = $\sum$counts[$\text{local}_L - 1$: |counts|] $\times$ c
15: **end while**
16: **if** $\text{local}_L < \text{score}_L$ **then**
17: $\text{score}_L = \text{local}_L$
18: **end if**
19: **end for**
20: **return** score_L

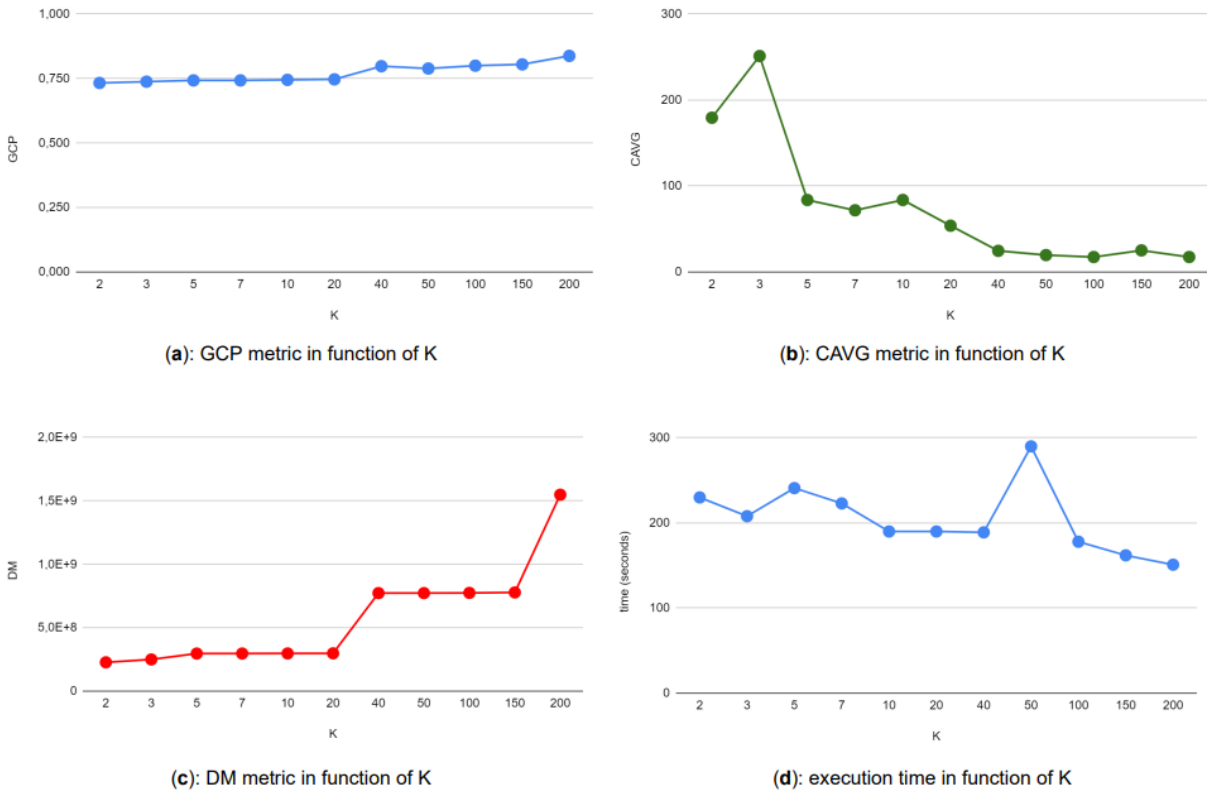
is checked once more, until it reaches 0. The computational cost of L-Recursive is $O(|P| \times |s|)$, with P being the number of Equivalence Classes and $|s|$ the length of the set of sensitive values in the sensitive attribute.

Evaluation 5

Evaluating our tool consists on using the different metrics explained in previous sections to analyze the results for different degrees of K and L. We aim to understand the underlying effect of increasing the generalization of a dataset and observe how it affects the security of the individuals within the dataset, as well as the data's usefulness.

5.1 Evaluation for levels of K

K-Anonymization is the first step in generalizing a dataset, therefore we want to see its effects on information loss, the dataset's discernibility and the average equivalence class size.



(a): GCP metric in function of K

(b): CAVG metric in function of K

(c): DM metric in function of K

(d): execution time in function of K

Figure 5.1. Metrics for different levels of K. In (a) we find the GCP metric, (b) the CAVG metric, (c) the DM metric and finally (d) the execution time of the algorithm.

Its performance regarding to the evaluation metrics are found in Figure 5.1, for different levels of K.

In Figure 5.1(a), we observe the results of Information Loss for different levels of K. As explained in chapter 3, the Information Loss metric returns a value closer to 0 when the generalized dataset is similar to the original microdata, while a value closer to 1 would signify a completely generalized dataset. For that reason, it makes sense to notice an increase in the metric for greater values of K, as the generalization is greater. It is also noticeable that for $K = 2$, the metric already returns a high GCP. Its cause is that the algorithm that this paper proposes ensures anonymization for K, while other tools do not necessarily force all equivalence classes to have a cardinality of K or more. Therefore, if making a partition means that at least the size of one equivalence class is below K, then the dimension by which the partition was made will not be able to split again. This decreases the number of partitions, therefore makes the data less useful, but ensures the individuals' privacy. Moreover, this metric depends greatly on the input dataset: whereas a dataset with less attributes and simple taxonomy trees could generally make simple partitions, the Adult dataset contains a considerable amount of attributes and uneven distributions. All in all, the results of the GCP metric describe a satisfactory level of security and privacy for the individuals in the dataset due to generalized data, but less useful for data analysis.

The Average Equivalence Class Size Metric describes the amount of equivalence classes compared to how many there should be for a value K. Contrarily to the other metrics, CAVG decreases when the value of K increases. This is explained by the fact that, the higher the value of K, the less Equivalence Classes are going to be generated and the less penalty they will receive due to an increased K. The reason for a lower amount of Equivalence Classes is that, since the algorithm ensures that every Equivalence Class has at least size K, generalization remains high and less partitions can be generated. Therefore, the algorithm seems to create too few equivalence classes for a smaller K, something that can be gradually fixed by increasing it.

In Figure 5.1(c) The Discernibility Metric describes the size of the generated equivalence classes. An equivalence class with a greater size will return a greater DM value. Similarly to GCP, for lower values of K, the algorithm returns smaller values. This is explained for the reason that aiming for a greater K value will return larger equivalence classes. Regardless, for every K value the returned value is considerably big. As mentioned in GCP, since the algorithm does not allow for a single equivalence class size to go under K, fewer partition possibilities can be performed. Moreover, this metric depends on the order that the dimensions were split, since the very first partition will create equivalence classes that the following partitions will have to consider while satisfying the anonymization properties. Therefore the metric could return lower values if we modified the heuristic that decides the order in which dimensions are split.

In conclusion for the evaluation of different levels of K, it is clear that not allowing a single record to fall under K highly restricts data usefulness. However, it successfully protects the privacy of each individual which is a key aspect in anonymization. We have also seen how certain metrics correlate to eachother: when DM reports sizeable equivalence classes, CAVG describes that not

enough equivalence classes have been created and viceversa. Both go hand in hand as a very populated equivalence class decreases the amount of other possible equivalence classes.

5.2 Evaluation for levels of L

Since L-Diversity builds on top of K-Anonymization, and we have seen how the different metrics perform under different levels of K, we can analyze the algorithm for different levels of L as well. For that, we have picked a fixed K level. This K level has to be sufficiently large to

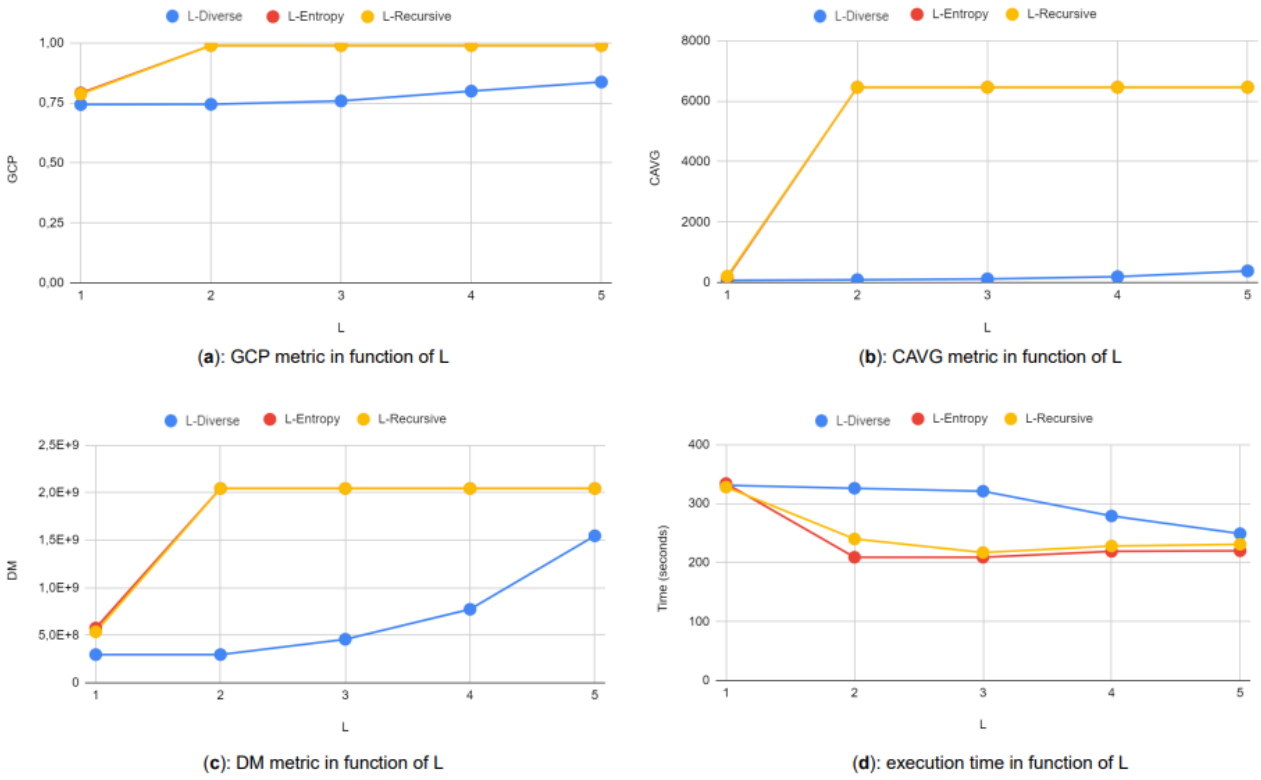


Figure 5.2. Metrics for different levels of L. For each, we can find the different types of L-Diversity. In (a) we find the GCP metric, (b) the CAVG metric, (c) the DM metric and finally (d) the execution time of the algorithm.

In Figure 5.2 we find the different results obtained for each L-Diversity definition, for different levels of L.

Figure 5.2(a) gathers the evaluation of the Information Loss metric, and we can immediately observe that L-Diverse obtains a less generalized dataset, while L-Entropy and L-Recursive have very similar outputs. The reason why L-Diverse obtains lower Information Loss is because it is much less restrictive than its counterparts. The only condition for L-Diverse to be satisfied is that the amount of sensitive values within each Equivalence Class is at least L. However, even with low restrictions, the metric obtained is quite high, with an average of 0.77. The reason for this is seen in Section 3.1, where we observed how the sensitive attribute "Race" has extremely

unbalanced sensitive values, giving rise to less Equivalence Classes being able to contain the other less-represented sensitive values. While this makes the anonymization of the dataset less useful, it also informs the dataset owner that the other sensitive values should be better represented for a preferable anonymization to be performed. As for the other metrics, since they are more restrictive than L-Diverse, they have an upperbound of 0.989. Finally, it is clear that increasing the degree of L will lead to more information loss, and therefore, a more protected and secure dataset.

As for Figure 5.2(b), we can observe that L-Entropy and L-Recursive obtain very large and similar results, with an average of 6460 for $L > 1$. Contrarily, L-Diverse has an average of 200 for $L > 1$. The reason the values of L-Diverse is so low is due to the fact it is less restrictive than the other two definitions of L-Diversity, as a satisfactory of Equivalence Classes have been created to represent the dataset, as opposed to L-Entropy and L-Recursive which generate a fewer amount of Equivalence Classes and therefore cannot properly anonymize the dataset.

Similarly, Figure 5.2(c) shows a small discernibility degree for L-Diverse which slightly increases as the degree of L becomes larger. Just like in K-Anonymization, this metric goes hand in hand with the Average Equivalence Class Metric, as a fewer amount of Equivalence Classes will generally result in an uneven distribution among them, especially when the sensitive attributes are not well distributed. Regarding L-Entropy and L-Recursive, the same conclusions as with the previous metrics can be drawn, as they are more restrictive definitions for managing sensitive values within Equivalence Classes.

In conclusion for the metrics concerning L-Diversity, we have seen how L-Diversity adds a new layer of security: while K-Anonymization would return an average degree of Information Loss since its properties were satisfied, L-Diversity makes sure that the sensitive values are "well-represented", not only among the Equivalence Classes, but also in regards to their distributions.

5.3 Time complexity

The execution time of the algorithm remains standard, with an average of 3 minutes to generalize the Adult dataset. However, it is important to note that the amount of time taken greatly depends not only on the number of individuals present in the dataset, but also the amount of attributes and whether they are numerical or categorical. For numerical attributes, a bigger amount of candidates generated by the candidate function will increase the time of the algorithm, as partitions have to be made for each candidate as well as checking the anonymization properties. Meanwhile, for categorical attributes, an increased execution time depends on the taxonomy trees for each attribute. If a partition concerning a certain node in a tree has N children, and the anonymization properties are satisfied, then N new dimensions are created and have to go through the same process as its parent.

Regarding the results obtained, in Figure 5.1(d) we can observe that the amount of time taken by the algorithm decreases when the value of K increases. This is due to the same reason as the evolution of the other metrics: with a greater K, less partitions can be made and less dimensions are created.

Meanwhile, in Figure 5.2(d), we find the different execution times for each type of L-Diversity definition, in fuction of the degree of L. As it can be observed, L-Diverse takes more execution time than L-Entropy and L-Recursive. As pointed by the metrics in function of L, L-Diverse is the least restrictive concept between the three, therefore being able to make more partitions in the dataset, and as such, extending the execution time. Regarding the other two definitions of L-Diversity, L-Entropy seems to take less time than L-Recursive. However, as for the resulting metrics, both concepts obtain the exact same results. Therefore the reason why L-Entropy takes longer is because the complexity of the algorithm is larger.

Conclusion 6

In this work we have explored different types of anonymization tools and concepts, and how they can be considered a reliable technique to anonymize a dataset in order to make its data useful, and at the same time secure for the individuals. We have also explored possible options of what could be contained in a dataset, such as numerical and categorical data, and how this has an impact on which techniques need to be used in relation to the type. We have added different types of tools to allow anonymization, as well as added different layers, in the form of L-Diversity and its branches, which are necessary to remove the vulnerabilities that K-Anonymization has. In addition, we have evaluated the resulting dataset with a series of metrics that not only indicate the amount of information hidden in the end result, but also gives insights on how the process of anonymizing a dataset was performed, by giving details on the amount of Equivalence Classes and its distribution. With these insights we can better fine tune the algorithms by adjusting the values assigned for the techniques to help optimise and ensure better privacy for the data, while attempting to retain its usefulness.

The use of the anonymization properties, mainly Entropy L-Diversity and Recursive L-Diversity, can be considered a potential method to anonymize a dataset as long as the data within it is in satisfactory conditions. A sensitive attribute with poorly represented values will not satisfy the L-Diversity properties.

The tool currently consists of an algorithm that is able to take a dataset and refine it to the expected levels of anonymization. To ensure a full fledged tool that can be utilised, the following lines of work may be considered. Firstly, to carry out a more extensive exploration of the different tools and concepts, since solving a poorly represented sensitive value is not always possible. For this reason, we introduce concepts such as Differential Privacy, which can benefit the anonymization procedure. For large datasets, Differential Privacy can ensure that the privacy for each individual is upheld by adding noise into the output dataset, which is controlled by a chosen epsilon value, without disrupting the statistical value. Secondly, add another layer to the procedure called T-Closeness, which can solve some of the vulnerabilities caused by L-Diversity. Thirdly, ensure that there is a user-friendly interface for the tool.

Bibliography

[1] Boris Lubarsky. Re-identification of "anonymized" data. *Georgetown Law Technology Review. Available online: https://www. georgetownlawtechreview. org/re-identification-of-anonymized-data/GLTR-04-2017 (accessed on 10 September 2021)*, 2010.

[2] Jan Philipp Albrecht. How the gdpr will change the world. *Eur. Data Prot. L. Rev.*, 2:287, 2016.

[3] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient k-anonymization using clustering techniques. In *International Conference on Database Systems for Advanced Applications*, pages 188–200. Springer, 2007.

[4] Tochukwu Iwuchukwu, David J DeWitt, AnHai Doan, and Jeffrey F Naughton. K-anonymization as spatial indexing: Toward scalable and incremental anonymization. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1414–1416. IEEE, 2007.

[5] Sean Chester, Bruce M Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo, and S Venkatesh. k-anonymization of social networks by vertex addition. *ADBIS (2)*, 789:107–116, 2011.

[6] Datatilsynet. Afgørelse om brug af google analytics fra det østrigske datatilsyn. Accessed: 03/10-2022.

[7] PWC. Sådan beregner datatilsynet gdpr-bøder. Accessed: 03/10-2022.

[8] Lars Mathiassen, Andres Munk-Madsen, Axel Nielsen, and Jan. Stage. *Object Oriented Analysis Design.* Metodica ApS, 2018.

[9] Hongwei Tian and Weining Zhang. Extending -diversity to generalize sensitive data. *Data & Knowledge Engineering*, 70(1):101–126, 2011.

[10] Amin Aminifar, Fazle Rabbi, Violet Ka I Pun, and Yngve Lamo. Diversity-aware anonymization for structured health data. pages 2148–2154, 2021.

[11] Vanessa Ayala-Rivera, Patrick Mcdonagh, Thomas Cerqueus, and Liam Murphy. A systematic comparison and evaluation of k-anonymization algorithms for practitioners. *Transactions on Data Privacy*, 7:337–370, 12 2014.

[12] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. Fast data anonymization with low information loss. In *Proceedings of the 33rd international conference on Very large data bases*, pages 758–769, 2007.

[13] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Multidimensional k-anonymity. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.