UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
DEPARTMENT OF MATHEMATICS AND INFORMATICS

Data mining with privacy guarantees: Decision trees with differential privacy

-MASTER THESIS-

Author:
Jelena Novaković

Mentor:
Dragana Bajović, PhD

Novi Sad, September 2019

Declaration of Authorship

I, Jelena Novaković, declare that this thesis titled, "Data mining with privacy guarantees: Decision trees with differential privacy" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Acknowledgement

I would like to express my deep, sincere gratitude to my mentor Dragana Bajović for her patience, motivation, unending support and most importantly for many insightful conversations we had during the development of this master thesis.

This thesis was done in the framework of European Union's Horizon 2020 Research and Innovation project I-BiDaaS, grant agreement No. 780787.

Data mining with privacy guarantees

Decision trees with differential privacy

Author: Jelena Novaković

Abstract

With the rise of Internet of Things where various devices, sensors, software, etc. are collecting all kinds of data about our physical world, including humans, privacy preservation is becoming increasingly challenging. The European Union's General Data Protection Regulation (GDPR) that came into force on May 25$^{\text{th}}$, 2018, aims to give control to the citizens regarding the use of their personal data by businesses and enterprises, by defining requirements for processing this kind of data. Specifically, the GDPR requires that any business process must implement data protection by design and by default.

One principled approach to achieve this in the context of data mining is the mechanism of *differential privacy*. With differential privacy, there is a guarantee that individual records of a dataset cannot be learned even when an arbitrary external information is provided. This is achieved by introducing randomness, by the data curator, into response to queries, posed by machine learning algorithms to the database. The size of the randomness is controlled by the, so called, epsilon factor, or the privacy budget that trades-off the level of achieved privacy with performance of the machine learning algorithm in question.

Depending on a given machine learning algorithm, there are many ways in which one can implement differential privacy. In practice, the latter translates into the question at what level of the algorithm and what kind of queries will be perturbed by noise.

This master thesis aims at surveying different differential privacy schemes and the resulting system architectures. Specific examples that will be used for illustration of general principles for query perturbation will include decision tree algorithm, i.e., two commonly used differentially private decision tree algorithms will be described theoretically and implemented on real datasets.

Contents

1 Introduction **10**

2 Privacy preservation techniques **12**
2.1 Anonymization 12
2.2 K-anonymity, L-diversity and T-closeness 13

3 Differential privacy **16**
3.1 Preliminaries 16
3.2 Definition of differential privacy 17
3.3 Differentially private mechanisms 19
3.3.1 Laplace mechanism 19
3.3.2 Exponential mechanism 21
3.4 Composition theorems 23
3.4.1 Sequential composition 23
3.4.2 Parallel composition 24

4 Data mining with differential privacy **26**
4.1 Interactive approach 26
4.2 Non-interactive approach 27
4.2.1 Central privacy 28
4.2.2 Local privacy 28
4.3 Privacy vs. utility 29
4.4 Choosing the privacy budget ε 30

5 Decision tree algorithm **32**
5.1 Motivation 32
5.2 Creating a tree 32
5.3 Choosing a value for splitting 34

6 Differentially private decision tree algorithm **36**
6.1 Differentially private mechanisms within decision tree algorithm 36
6.2 Dividing a privacy budget 37
6.3 Creating a private tree 38
6.4 Quality functions 41
6.4.1 Max operator 41
6.4.2 Gini index 42
6.5 Stopping criteria 42
6.6 Implementation in Python 43

7 Experimental results **52**
7.1 Experimental setup 52
7.2 Programming language 52
7.3 Metrics, cross–validation and privacy budget 53
7.4 Datasets 53
7.5 Results 55

8 Conclusion **62**

9 Appendix **64**

10 List of Figures **67**

11 List of Listings **68**

12 List of Algorithms **69**

1 Introduction

Historically, privacy was almost implicit, because it was hard to find and gather information. But in the digital world, whether it's digital cameras or satellites or just what you click on, we need to have more explicit rules - not just for governments but for private companies.

Bill Gates

In the past decades, private companies, governments, hospitals, social networks etc. have been collecting vast amount of digitized personal information about the individuals who are theirs clients, customers or patients. Collected data created opportunities for improvement of human well-being in various ways. For example, medical data can be used and analysed in order to prevent epidemics, discover hidden connections between diseases, track the spread of a disease, etc. While it is very useful to gain knowledge from the data, it is also important to preserve the privacy of the individuals who are participating in the research. Since the collected data may be highly sensitive in terms of privacy (e.g., personal information like name, surname, tax and national ID number, address, etc.), data owners must find a way to protect the data from any attack.

In this thesis, we will describe one solution that preserves data privacy. The solution relies on the implementation of differential privacy within machine learning algorithm decision tree.

The thesis consists of 8 chapters organized as follows. Chapter 2 - *Privacy preservation techniques* reviews the related work on the privacy preservation methodologies. Chapter 3 - *Differential privacy* explains in more details the main concept of Differential Privacy, its properties and mechanisms. Chapter 4 - *Data mining with differential privacy* describes two commonly used approaches for applying differential privacy in data mining. Chapter 5 - *Decision tree algorithm* reviews the basic workflow of decision tree algorithm in non-private scenario. Chapter 6 - *Differentially private decision tree algorithm* introduces two versions of differentially private decision tree algorithm, each covering first the theoretical background and then implementation in Python. Chapter 7 - *Experimental results* explains testing setups and discusses the results. Chapter 8 - *Conclusion* summarizes the whole work and proposes ideas and directions for future work and improve-

ments. Finally, in *Appendix* we added Python script developed for testing the presented algorithms.

2 Privacy preservation techniques

In this chapter we will review some of the most commonly used privacy preservation techniques developed before differential privacy. We will briefly explain how they work and why they were not sufficient for privacy protection.

2.1 Anonymization

Personally identifiable information (PII) is any data that can be used to identify individuals, either directly or by combining the information with other data. For example, name, social security number and mobile phone number of an individual are all instances of PII.

Data anonymization is the process of either encrypting or removing personally identifiable information from a dataset in order to protect the privacy of the individuals. The main goal of the anonymization is to ensure that any individual's PII will not be disclosed by releasing the data and yet to provide valuable data for the analysis.

Even though it sounds like the process of anonymization should perfectly protect individual's privacy, in 1997 Latanya Sweeney, then MIT graduate student in computer science, demonstrated the opposite [1], [2]. She had shown that the information about the individuals in in the published dataset with identifying information removed, could still be uniquely re-identified. She also found that 87% of the American population can be uniquely identified by date of birth, gender and postal code[1].

[1]Sweeney's studies in computational disclosure control had brought to her attention hospital records data released to researchers by the Massachusetts Group Insurance Commission (GIC) for the purpose of improving healthcare and controlling costs [2]. At the time GIC released the data, William Weld, the Governor of Massachusetts, assured the public that GIC had protected patients' privacy by deleting personally identifiable information of the patients. In order to demonstrate the opposite, Sweeney started hunting for the Governor's hospital records in the GIC data (she knew that he had collapsed in public earlier that year). For twenty dollars, she purchased the complete voter rolls from the city of Cambridge. The purchased database contained, among other things, the name, address, ZIP code, birth date, and gender of every voter. By combining this data with the GIC records, Sweeney easily found Governor Weld. Only six people in Cambridge shared his birth date, only three of them were men, and only he lived in his ZIP code. In a theatrical flourish, Dr. Sweeney sent the Governor's health records to his office [2].

This identity disclosure motivated scientists to search for better solutions for privacy preservation that simple identity suppression. The following subsection introduces three more advanced privacy protection techniques: *K-anonymity*, *L-diversity* and *T-closeness*.

2.2 K-anonymity, L-diversity and T-closeness

K-anonymity

The idea of K-anonymity is to protect the privacy of the data by using quasi-identifiers.

Quasi-identifiers are the attributes that can be used jointly to identify a person or a group of persons, such as gender, ZIP code, profession, birth date, race, etc. In the Sweeney example, neither birth date, nor ZIP code or sex could identify an individual, but thier combination is likely to achieve this.

K-anonymity is a privacy property requiring that all combinations of quasi-identifiers in a database are repeated for at least k individuals [3]. This technique was proposed by Samarati and Sweeney [1] and it guarantees that whenever an attacker uses a quasi-identifier to attack a user, he/she will always obtain at least K similar candidates [30]. Hence the probability of re-identification of a particular individual in the database is $\frac{1}{K}$.

K-anonymity provides stronger privacy protection than anonymization, but it still has some weaknesses. For example, if an attacker wants to find out whether or not a certain individual in a K-anonymous dataset has cancer and the group of individuals with the same quasi-identifiers (e.g., birth year and city) all have cancer, it is obvious that the privacy of the information is still compromised. If the attacker knows the individual's birth year and city, even though the observed dataset is K-anonymous, he can easily conclude that the individual has cancer, because he knows that all individuals in the dataset that are born in the same year and living in the same city as the observed individual are ill. This example shows us that K-anonymity can create groups that leak information due to lack of diversity in the sensitive attribute.

L-diversity

The L-diversity models are proposed to deal with the drawbacks of K-anonymity technique. L-diversity requires the sensitive information in an

anonymous group, i.e., group of individuals with the same quasi-identifiers, must have enough "diversity" [30]. More formally, a dataset is said to satisfy L-diversity if, for each group of individuals sharing a combination of quasi-identifiers, there are at least L "well-represented" values for each confidential attribute [3].

T-closeness

T-closeness requires the distribution of sensitive information in any anonymous group must be close enough to the distribution of the whole dataset [30]. A data set is said to satisfy t-closeness if, for each group of individuals sharing a combination of quasi-identifiers, the distance between the distribution of the confidential attribute in the group and the distribution of the attribute in the whole data set is no more than a threshold t [3].

The biggest weakness of the privacy protection techniques described in this subsection is the need of predefining the background knowledge of the attackers. In order to define quasi-identifiers, data owners must know the level of knowledge of the attackers, which is quite a strong requirement. If the background knowledge of the attackers is unknown, the protection may fail completely.

Unlike previous methods for privacy protection, mechanisms of differential privacy provide a powerful privacy protection which does not require any insights into the structure of knowledge of the attackers. The next chapter reviews differential privacy mechanisms.

3 Differential privacy

The main goal of data analysis and machine learning is to extract useful information about the data, such as how to find clusters of similar samples, how to predict a certain quantity or how to classify the data. The output model of machine learning algorithms can be used to represent the distribution of the analysed data or to deal with future data. Therefore, the model encodes information about analysed data.

On the other hand, the goal of privacy preservation in data mining is to protect the privacy of the data. It is obvious that machine learning and privacy preservation are in opposition, but it is not impossible to achieve both. The solution for this problem is to make machine learning more privacy aware and to find optimal trade-off between the privacy protection and data analysis. The objective of privacy aware data analysis is to discover useful information without sacrificing the privacy of any individual in the database.

Differential privacy is a powerful definition for privacy preservation proposed by Cynthia Dwork, and it is one of the most popular definitions of privacy today [4], [5], [6]. Google and Apple were the first two companies that implemented differential privacy in their businesses [18], [16]. Differential privacy is based on the idea that the outcome of the statistical analysis is essentially equally likely independent of whether any individual joins or refrains from joining the database, i.e., one learns approximately the same thing either way [11]. Differential privacy addresses the paradox of learning nothing about an individual while learning useful information about a population [4].

The following subsections formally define differential privacy and introduce commonly used methods and mechanisms for achieving differential privacy.

3.1 Preliminaries

First we will introduce some basic concepts and notations which will be helpful later.

Let D be the dataset with n samples, where each sample has d features.

Neighboring datasets: Two datasets D_1 and D_2 (subsets of D) are

called neighboring if they have the same cardinality but differ in only one record.

Query: A query q is a mapping that maps dataset D to the set of real numbers, i.e., $q : D \mapsto \mathbb{R}$. A group of queries is denoted by $\mathcal{Q}$.

Differential privacy provides *a randomized mechanism* $\mathcal{M}$ to mask the difference of query q between two neighboring datasets D_1 and D_2 [15]. The maximal difference on the results of query q is known as the l_1 *sensitivity* of the query q. l_1 sensitivity is then used to define how much perturbation is required in order to preserve privacy of the desirable answer. The randomized mechanism $\mathcal{M}$ can be consider as an algorithm which accesses the dataset and performs some functionality, while ensuring required level of privacy. For example, if we add noise σ to the query result $q(D)$, then the noisy answer $\hat{q}(D)$ denotes the randomized answer of query q.

Randomization is an essential element of privacy preservation based on differential privacy. Actually, any non-trivial privacy guarantee that holds regardless of all present or even future sources of auxiliary information, including other databases, studies, Web sites, online communities, gossip, newspapers, government statistics, and so on, requires randomization [4]. And why is that? Well, let us suppose the contrary. Suppose that there exists a query q and two datasets D_1 and D_2 that yield different outputs, q_1 and q_2, under the query q. Changing one record at a time we see there exists a pair of datasets differing in only one record, on which the same query yields different outputs, and knowing that the dataset is one of these two almost identical datasets, an adversary learns the value of the data in the unknown record. By masking the difference of the outputs q_1 and q_2, differential privacy promises that an adversary learns nothing more about an individual record regardless of its present in the dataset. The next subsection gives the formal definition of differential privacy.

3.2 Definition of differential privacy

Differential privacy gives a promise that releasing the aggregated results should not reveal too much information about any individual record in the dataset. More formally:

Definition 1 ((ε, δ) - *differential privacy*) [4].
A randomized mechanism $\mathcal{M}$ provides (ε, δ) - differential privacy if for

all $\Omega \subseteq \ Range(\mathcal{M})$ and for any pair of neighboring datasets D_1 and D_2, $\mathcal{M}$ satisfies:

$$Pr[\mathcal{M}(D_1) \in \Omega] \leq exp(\varepsilon) \cdot Pr[\mathcal{M}(D_2) \in \Omega] + \delta.$$

For $\delta = 0$, $\mathcal{M}$ provides ε-*differential privacy*. More formally,

Definition 2 (ε - *differential privacy*) [7].

A randomized mechanism $\mathcal{M}$ provides ε - differential privacy if for all $\Omega \subseteq \ Range(\mathcal{M})$ and for any pair of neighboring datasets D_1 and D_2, $\mathcal{M}$ satisfies:

$$Pr[\mathcal{M}(D_1) \in \Omega] \leq exp(\varepsilon) \cdot Pr[\mathcal{M}(D_2) \in \Omega].$$

The ε-differential privacy provides stronger privacy guarantee than the (ε, δ) - differential privacy. Because of that, for the rest of the thesis, we will focus on the ε-differential privacy.

Let us explain the previous definition in more details. From Definition 1, for $\delta = 0$ we have:

$$\frac{Pr[\mathcal{M}(D_1) \in \Omega]}{Pr[\mathcal{M}(D_2) \in \Omega]} \leq exp(\varepsilon).$$

Furthermore, considering that we can exchange D_1 and D_2, Definition 1 implies:

$$exp(-\varepsilon) \leq \frac{Pr[\mathcal{M}(D_1) \in \Omega]}{Pr[\mathcal{M}(D_2) \in \Omega]} \leq exp(\varepsilon).$$

Finally, since $exp(\varepsilon) \approx 1 + \varepsilon$, for ε small enough, we have:

$$1 - \varepsilon \leq \frac{Pr[\mathcal{M}(D_1) \in \Omega]}{Pr[\mathcal{M}(D_2) \in \Omega]} \leq 1 + \varepsilon.$$

So, Definition 1 claims that, for ε small enough, the probability that the output of the mechanism $\mathcal{M}$ over dataset D_1 falls into the set of outcomes Ω is nearly the same as the probability that the output of the mechanism $\mathcal{M}$ over dataset D_2 falls into the same set of outcomes. Thus, as it is presented in Figure 1, differential privacy ensures that observing the output of mechanism M over the two neighboring datasets, an adversary learns nothing more about an individual record regardless of whether the record is present or absent in the analysis.

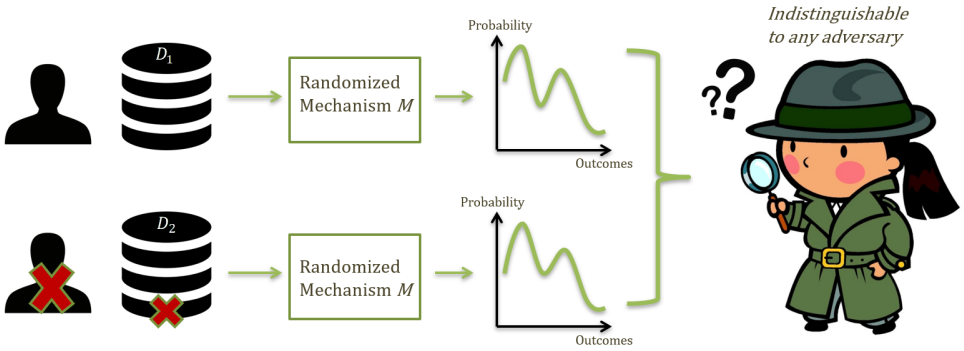

Figure 1: *Definition of Differential Privacy [46], [47], [49]*

Parameter ε from Definition 1, is called *the privacy budget* [15]. Privacy budget controls the level of privacy of the mechanism $\mathcal{M}$. Smaller values of ε imply stronger privacy. After introducing differentially private mechanisms, we will review commonly used properties of the privacy budget in subsection *3.4 Composition Theorems*.

3.3 Differentially private mechanisms

After formal definition of differential privacy, we will explain in which ways differential privacy can be achieved in practice.

There are three widely used mechanisms that achieve differential privacy: *Laplace mechanism*, *Gaussian mechanism* and *Exponential mechanism*. In our work we implemented Laplace and exponential mechanisms, which we will now define.

3.3.1 Laplace mechanism

Laplace mechanism is used for numeric types of queries. For example, if one were to apply a query that counts the number of records in a dataset, then the Laplace mechanism should be used. The mechanism is very simple. It consists of adding Laplacian noise to the answer of query q before returning the result to the data analyst (i.e., data mining algorithm). The addition of noise is sampled from the Laplace distribution shown in Figure 2, and the scale of noise is calibrated to the sensitivity of query q divided by ε.

Before we give a formal definition of Laplace distribution and Laplace

mechanism, we will introduce l_1 sensitivity - one of the important parameters that determines how accurately we can answer numeric queries.

Definition 3 (l_1 *sensitivity*) [4].
The l_1 sensitivity of a function $f : D \mapsto \mathbb{R}$ is:

$$\Delta_f = \max_{\substack{D_1, D_2 \subseteq D \\ \|D_1 - D_2\|_1 = 1}} \|f(D_1) - f(D_2)\|_1.$$

The l_1 sensitivity of a function f is the parameter that defines the amount of perturbation required in the differentially private mechanisms. It captures the magnitude by which a single individual's data can change the function f in the worst case, and therefore, the uncertainty in the response that we must introduce in order to hide the participation of an individual. The l_1 sensitivity of a function is defined, in a sense, for the worst case neighboring database pair and because of that it gives an upper bound on how much we must perturb its output to preserve privacy [4].

Definition 4 (*The Laplace Distribution*) [4].
The Laplace Distribution (centered at 0) with scale b is the distribution with probability density function:

$$Lap(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

The variance of this distribution is $\sigma^2 = 2b^2$. We will write $Lap(b)$ to denote the (0-centered) Laplace distribution with scale b.

The Laplace mechanism is formally defined as:

Definition 5 (*The Laplace Mechanism*) [15], [8].
Given a function $q : D \mapsto \mathbb{R}$ over a dataset D, the Laplace mechanism, denoted by M_L, is defined as:

$$M_L(D, q, \varepsilon) = q(D) + Lap\left(\frac{\Delta_q}{\varepsilon}\right),$$

where Δ_q denotes the l_1 sensitivity of query q.

It is easy to show that Laplace mechanism, $M_L(D, q, \varepsilon)$, satisfies the following property.

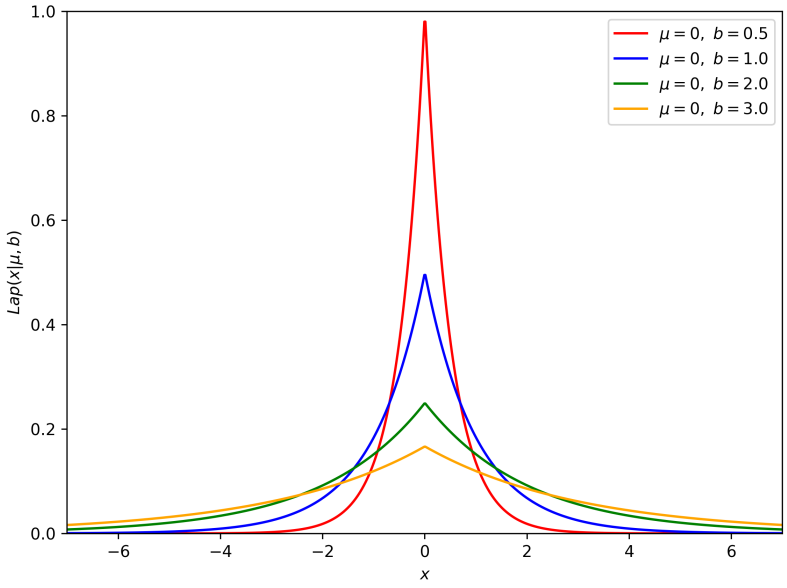

Figure 2: *Laplace distribution*

Theorem 1. The Laplace mechanism preserves ε-differential privacy [4].

The Gaussian mechanism has quite similar principle as the Laplace mechanism. It works in analogous way, except that the noise is now sampled from the Gaussian distribution instead of Laplacian.

3.3.2 Exponential mechanism

Even though the Laplace mechanism is a powerful method that ensures differential privacy, it can not be applied to non-numeric (categorical) queries. For that reason, the exponential mechanism has been proposed. The exponential mechanism represents another way to implement differential privacy and it is suitable for categorical queries. It was designed for situations in which we wish to choose the "best" response but adding noise directly to the computed quantity can completely destroy its value. Example include setting a price in an auction where the goal is to maximize revenue. If we add the small amount of Laplacian noise to all of the observed prices (in order to protect the privacy of a bid), before choosing the best one (the one that maximizes revenue) we could dramatically reduce the resulting revenue [4]. The

exponential mechanism is the natural building block for answering queries with arbitrary utilities (and arbitrary non-numeric output), while preserving differential privacy.

Given some arbitrary range $\mathcal{R}$, the exponential mechanism is defined with respect to some utility (quality) function $u : D \times \mathcal{R} \mapsto \mathbb{R}$, which maps dataset-output pairs to utility scores. Intuitively, for a fixed dataset D, the user prefers that the mechanism outputs some element of $\mathcal{R}$ with the maximum possible utility score. The sensitivity of the utility score u, Δ_u, is defined as the largest possible difference in the utility score when applied to two inputs that differ only on a single record, for all r [12].

$$\Delta_u = \max_{r \in \mathcal{R}} \max_{\substack{D_1, D_2 \subseteq D \\ \|D_1 - D_2\|_1 = 1}} \|u(D_1, r) - u(D_2, r)\|_1 \text{ [4].}$$

The intuition behind the exponential mechanism is to sample the outcome from the probability distribution induced over the output domain of the utility function u, i.e., to output each possible $r \in \mathcal{R}$ with probability proportional to $\exp \varepsilon(\frac{u(D,r)}{2\Delta_u})$. Mechanism gives strong utility guarantees, because it discounts outcomes exponentially quickly as their quality score falls off.

The exponential mechanism is formally defined as:

Definition 6 (*Exponential Mechanism*) [4].

The exponential mechanism $M_E(D, u, \mathcal{R}, \varepsilon)$ selects and outputs an element $r \in \mathcal{R}$ with the probability proportional to $\exp \varepsilon(\frac{u(D,r)}{2\Delta_u})$.

The following property holds.

Theorem 2. The exponential mechanism preserves ε-differential privacy [4].

More details about differentially private mechanisms as well as proofs of Theorem 1 and Theorem 2 can be found in [4].

One useful property of differential privacy that makes it much more practical is composability. Next subsection introduces two commonly used composition theorems.

3.4 Composition theorems

Now that we have several building blocks for designing differentially private algorithms, we want to understand how we can combine them to design more sophisticated algorithms. In order to use these tools, we would like that the combination of two differentially private algorithms be differentially private itself [4]. In this subsection we give theorems showing the parameter ε compose when differentially private algorithms are combined.

There are two basic privacy budget compositions which are widely used in practice: *the sequential composition* and *the parallel composition*. Sequential composition guarantees that a sequence of k mechanisms applied on the same dataset, where each of them ensures ε-differential privacy, will ensure $(k\cdot \varepsilon)$-differential privacy. Suppose now that we would like to apply a sequence of k mechanisms on a k disjoint subsets of a given dataset D, such that each of the k mechanisms is applied on exactly one subset of D. Parallel composition states that the final mechanism that includes the combination of k ε-differentially private mechanisms will also ensure ε-differential privacy. In other words, the privacy cost of multiple queries applied to the same data composes (i.e., is summed together), and a single query applied to different subsets of data (with no overlapping records) can be posed in parallel at no extra cost [34].

3.4.1 Sequential composition

Theorem 3 (*Sequential Composition*) [15], [14].

Suppose that a set of privacy mechanisms $\mathcal{M} = \{M_1, M_2, ..., M_k\}$ are sequentially performed on a dataset D, where each M_i provides ε_i-differential privacy. Then $\mathcal{M}$ will provide $(\varepsilon_1 + \varepsilon_2 + \cdots + \varepsilon_k)$-differential privacy.

Sequential composition helps us in scenarios where we have a series of randomized mechanisms performed sequentially on a dataset, i.e., it guarantees privacy for a sequence of differentially private computations. Property says that the privacy budget will be added up (or, alternatively, reduced from the total budget epsilon) for each step of mechanism $\mathcal{M}$.

3.4.2 Parallel composition

Theorem 4 (*Parallel Composition*) [15], [10].

Suppose we have a set of privacy mechanisms $\mathcal{M} = \{M_1, M_2, ..., M_k\}$. Furthermore, suppose that each M_i provides ε_i-differential privacy on disjointed subsets of the dataset D. Then the sequence of M_i will provide $\max\{\varepsilon_1, \varepsilon_2, ..., \varepsilon_k\}$-differential privacy.

The parallel composition property is used in cases where each M_i is implemented on disjointed subsets of a dataset D. The property says that the final privacy guarantee only depends on the largest privacy budget.

4 Data mining with differential privacy

Combining differential privacy with data mining and machine learning algorithms is not an easy task. There are various situations that can happen during differentially private data mining, so every single step should be predefined well. The goal is to find a trade-off between the privacy and the utility of the machine learning algorithm. Before we explain two basic approaches for achieving differentially private data mining, we will assume the existence of a trusted and trustworthy curator (data owner) who holds the data of individuals in a database D. The intuition is that each record of the database D contains the data of a single individual, and the privacy goal is to simultaneously protect every individual record while permitting statistical analysis of the database as a whole [4]. Data mining with differential privacy can be achieved with two basic approaches: *interactive approach* and *non-interactive approach* [4], which we explain next.

4.1 Interactive approach

Basic workflow of the interactive approach is shown in Figure 3.

As the title of this subsection indicates, this approach involves interaction between the data analyst and data owners from the start to the very end of the process. Only data owners have a permission to query the true database (i.e., raw data). Data analyst is posing different queries to the database and data owners return back a noisy response. Data owners should predefine the amount of privacy budget they are willing to spend on data analysis, which implies that data analyst should plan his steps of data exploration beforehand because he does not want to be left "broke" before the full analysis. Once he spends all of the privacy budget he has been given, he has to stop the interaction.

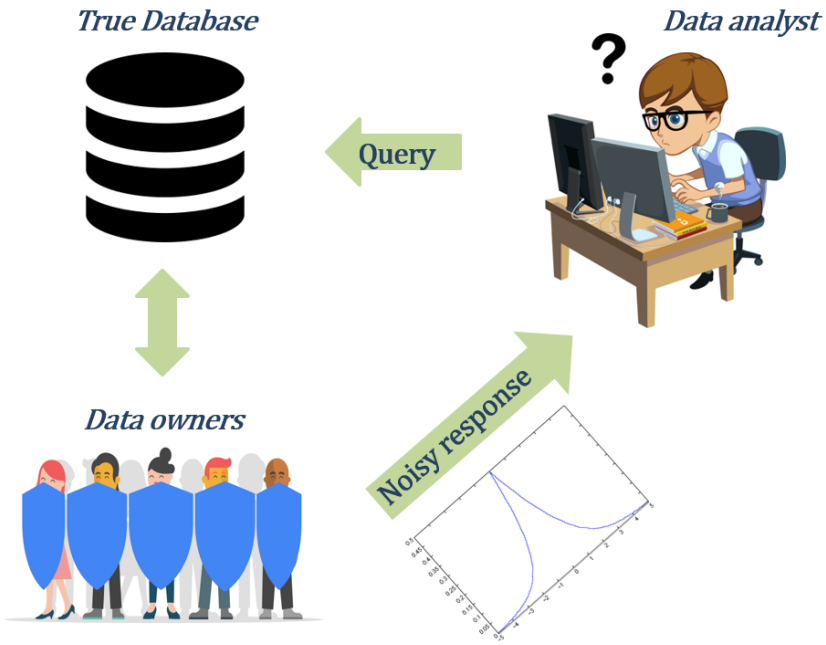

Figure 3: *Simplified scheme of Interactive approach [50], [51], [48]*

4.2 Non-interactive approach

Simplified scheme of non-interactive approach is shown in Figure 4. In contrast with the interactive approach where data owners should collaborate with data analyst along the whole process of any kind of statistical analysis over the true database, data owners publish a synthetic database. In this scenario anyone can have a permission to analyse the synthetic database and data owners do not need to worry about privacy preservation once they publish synthetic database.

Synthetic database can be developed from the true database in various ways. We will cover two most common designs: *central privacy* and *local privacy*.

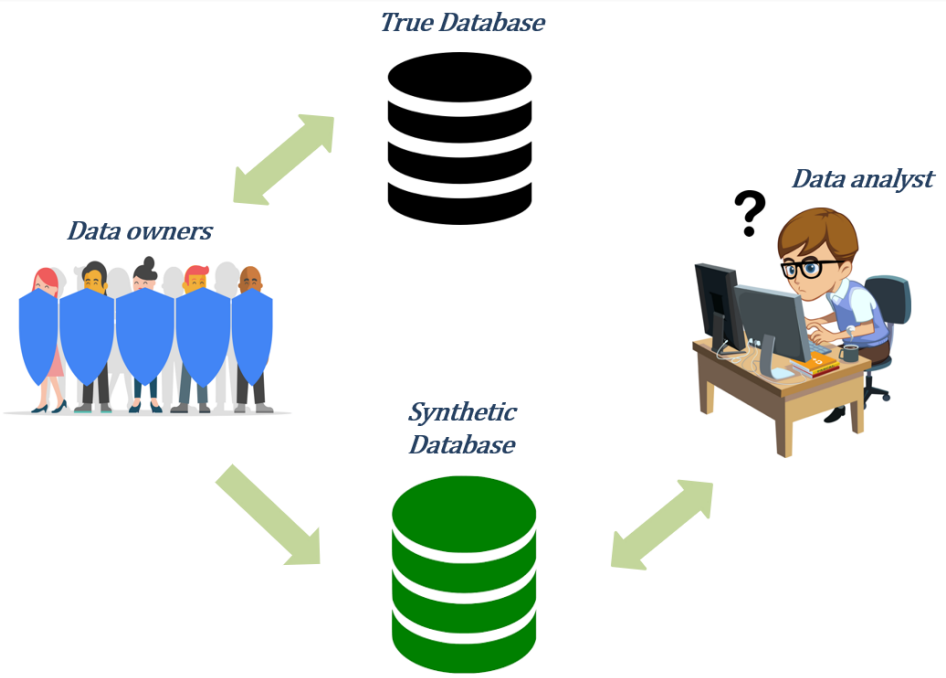

Figure 4: *Simplified scheme of Non-Interactive approach [50], [51], [48]*

4.2.1 Central privacy

Basic idea of central privacy is shown in Figure 5. Central Privacy requires from the data owners to collect raw data on the server first. True database D is stored in the data silos, after which the data is being manipulated in order to produce synthetic database D' which is not privacy sensitive at all. Once synthetic database has been produced, true database can be removed from the server forever.

4.2.2 Local privacy

A superior model - from the perspective of the owners of private data - would be a local model, in which agents could (randomly) answer questions in a differentially private manner about their own data, without ever sharing

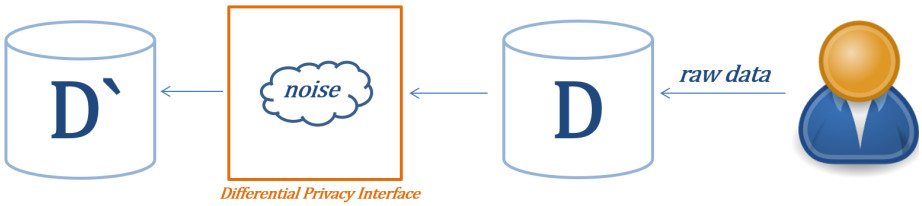

Figure 5: *Central Differential Privacy [52]*

it with anyone else [4]. So, unlike the previous model, in the local privacy scenario there is no trusted party at all. As it is presented in Figure 6, noise is added to the data even before it gets stored in the data silos, so no one has an insight into the true database D, i.e., only synthetic database D' is available for analysis. This kind of privacy preservation technique is commonly used for the Telecom operators, social networks and any kind of data that is collected through the internet.

For example, Apple is implementing local differential privacy techniques. They are using it for their Health application [17] and to calculate the frequency of used emojis in their applications. More about this work can be found in [16].

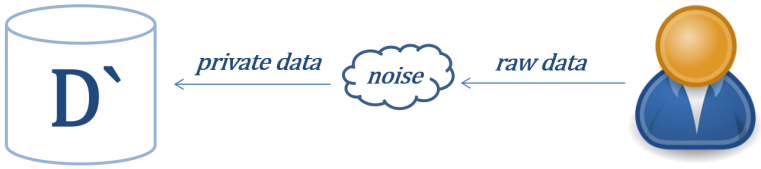

Figure 6: *Local Differential Privacy [52]*

4.3 Privacy vs. utility

While differential privacy is nice conceptually, it has been difficult to apply in practice. The parameters of differential privacy have an intuitive theoretical interpretation, but the implications and impacts on the risk of disclosure and choosing appropriate values for them is non-trivial.

A frequently asked question about both approaches is how much of a

privacy budget is enough to preserve the privacy of the true database and yet to allow the data analyst to gain useful insights from the data, i.e., how to determine the amount of ε required in order to protect the individuals privacy.

The answer is that it depends on many things, but it mostly depends on the domain of the analysed data and on the size of the dataset. If the domain of the data is highly sensitive (medical data, financial data, etc.), the privacy budget should be small, because there is more concern about privacy preservation. If the number of samples in the dataset is small, then the privacy budget should be large and vice versa.

Another question is which approach performs better and how to know which one is suitable for the certain data? The answer is the same as in the previous case, it all depends on the domain and specifications of the data.

4.4 Choosing the privacy budget ε

Despite the fact that the privacy budget ε is at the core of differential privacy mechanisms, very little work has been done in how to choose an appropriate value for ε in real world scenarios. Optimal ε should ensure good utility of machine learning algorithms, while providing an appropriate amount of privacy protection to each of the individuals in the dataset. To the best of our knowledge, a few papers have provided practical guidelines for choosing ε.

Vu and Slavkovic [19] provided guidelines for how to use hypothesis testing in a scenario where sensitive information needs to be protected. The main idea of this research was to try to estimate the additional number of records that are needed in order to produce the same quality of the results that the test would produce if privacy was not involved. They do so in terms of two main metrics: the confidence level of the hypothesis test (type I error) and the power of the hypothesis test (type II error) [33].

Lee and Clifton [20] proposed a model for calculating the optimal ε by considering an analyst as a Bayesian agent and observing a value of ε that controls how much the analyst's belief can change. They derived a bound of ε for the analyst's belief to remain below a given threshold.

He et al. [21] proposed an attack model that can provide the probability of success. They found that the upper bound of ε can be computed if the sensitivity Δ_f and length of the fault-tolerant interval are provided.

Justin et al. [22] proposed a principled economic approach for choosing the optimal ε.

5 Decision tree algorithm

Since the work presented in this thesis is based on the implementation of differentially private mechanisms within decision tree algorithm, here we will briefly introduce decision tree algorithm. First we will describe how it works in a non-private scenario, after which we will explain in which ways one can build differentially private decision tree.

5.1 Motivation

In machine learning, supervised learning is a process in which we are training our models (algorithms) to learn to classify data points correctly by feeding them with labeled data points. Decision tree is a commonly used supervised learning method for solving classification problems [23]. The main advantage of decision tree over other algorithms used for classification is that it is very explainable, it is a non-parametric algorithm, and it has the ability to reveal non-linear relationships among the features.

5.2 Creating a tree

Simply speaking, decision tree is an acyclic graph that can be used to make decisions [25]. There are different algorithms that can be implemented for creating a tree. In this subsection we will cover general principles underpinning decision tree algorithm.
Suppose we are given a dataset D with a set of attributes $\mathcal{A}=\{A_1, A_2, ..., A_d\}$ and a class attribute $C=\{c_1, \ c_2, ..., c_k\}$ that can take k different values. Each of the attributes in $\mathcal{A}$ can also take a number of values. Suppose further that we want to solve a multi-class classification problem over the dataset D with decision tree algorithm. Each record $s \in D$ has a class value c, from the class attribute C. The aim of a decision tree algorithm is to predict the class value for future unseen records s_{new}. We will denote with D_{c_l} the subset of dataset D, such that, for each $l = 1, 2, ..., k$, D_{c_l} contains records having the class attribute equal to c_l, and let D_c be a set that contains all $|D_{c_l}|$, if $D_{c_l} \neq \emptyset$, i.e., D_c contains the number of appearances of each class in dataset D. It is easy to see that the following equality

holds: $\sum_{l=1}^{k} |D_{c_l}| = |D|$[2].

Decision tree works recursively. In each iteration, the goal of any decision tree algorithm is to split the input dataset D in a way which will separate classes the best. The most common way is binary splitting into non-overlapping subsets (nodes) D_1 and D_2, which we also follow in this paper[3]. Algorithm chooses an attribute A from the set of attributes $\mathcal{A}$ and it chooses value v from all possible values of attribute A. If the attribute is numerical, algorithm should split D such that the values smaller than v fall into D_1 and values greater than v fall into D_2. Otherwise, if the attribute is categorical, algorithm puts all samples with the value of attribute A equal to v in D_1, and the rest in D_2[4]. After this, the whole procedure is done recursively on D_1 and D_2 by splitting them further with respect to new attributes, until the maximal depth is reached or predefined stopping criterion is satisfied. In each iteration the objective is to separate input dataset as "pure" as possible with respect to the class attribute C, that is, we seek for separations that yield child nodes (and eventually leaf nodes) as homogeneous as possible. Ideally, at the end of the procedure, we would like to have leaf nodes containing records representing just one class (if the latter is possible).

Algorithm 1 presents the general structure of the decision tree algorithm. The algorithm takes as input a dataset D, the set of attributes $\mathcal{A}$, the class attribute C, a maximal tree depth d, a quality function q and a parameter m_s representing minimal number of samples required in the training set in order to continue the algorithm, serving as a parameter for stopping criterion. The output of the algorithm is a decision tree model. Procedure `BuildTree` is responsible for node creation. In each iteration procedure `BuildTree` first checks if the stopping criteria are satisfied. If they are, the procedure returns

[2]Since D_{c_l} is a subset of D that contains all records from D, such that $s_c = c_l$, and since $\bigcap_{l=1}^{k} D_{c_l} = \emptyset$ (one record can not be labeled with two different classes), we can conclude that this equality holds. This conclusion will be important in subsection *6.3 Creating a private tree*.

[3]There exist also k-ary splittings, $k > 2$ [24], but they are less commonly used in practice.

[4]Suppose we are given a dataset D, where each record in D refers to an individual, and let the attributes of the dataset D be the following: *Name*, *Surname*, *Town* and *Gender*. Furthermore, suppose that the attribute chosen for the splitting attribute is a categorical attribute *Town*, that represents a town in which an individual is living. Finally, suppose that the attribute *Town* is taking three possible values : *'Beograd'*, *'Novi Sad'* and *'Kragujevac'*. If the value chosen for splitting is *'Beograd'*, then the algorithm will split dataset D on subsets D_1 and D_2, such that all records having the value of attribute *Town* equal to *'Beograd'* will fall into D_1, and all the rest records will fall into D_2.

a leaf node, and if not, it searches for the best value for further splitting. After finding the best value for splitting, procedure splits the dataset based on the chosen value and continues iteratively on created subsets.

Algorithm 1 Decision Tree algorithm

1: **procedure** DTree($D, \mathcal{A}, C, d, q, m_s$)
    Input: D - dataset, $\mathcal{A}$=$\{A_1,\ A_2$,..., $A_d\}$ - a set of attributes, C=$\{c_1,\ c_2$, ..., $c_k\}$ - a class attribute, d - maximal tree depth, q - quality function, m_s - minimal sample threshold
    BuildTree($D, \mathcal{A}, C, d, q, m_s$)
    return Decision Tree model
2: **end procedure**
3: **procedure** BuildTree($D, \mathcal{A}, C, d, q, m_s$)
4: **if** $\mathcal{A} = \emptyset$ or $d = 0$ or $|D| <= m_s$ or $|D_c| = 1$ **then**
    return a leaf labeled with $\text{argmax}_{c_l} D_{c_l}$
5: **end if**
    $(A_{opt}, v_{opt}) = \arg\max q\,(D, \mathcal{A})$
    $(D_1, D_2) = Split(D, A_{opt}, v_{opt})$
    BuildTree($D_1, \mathcal{A} \setminus A_{opt}, C, d-1, q, m_s$)
    BuildTree($D_2, \mathcal{A} \setminus A_{opt}, C, d-1, q, m_s$)
    return Decision node
6: **end procedure**

5.3 Choosing a value for splitting

Now the question is, how to choose this value v_{opt}? There are several quality functions that can be used for choosing value v_{opt}. The most commonly used is *Information Gain*, but in our work we focused on *Gini Index* and *Max Operator* because they are more adoptable to differential privacy frameworks (a broader explanation is given in subsection 6.4).

The job of the quality function q from Algorithm 1 is to decide which attribute and which value of that attribute will be used for splitting.

More formally, let q be a chosen quality function which will rank the values of the attributes. Also, let (A_{opt}, v_{opt}) be the solution of the optimization problem:

$$\arg\max_{A \in \mathcal{A}} \arg\max_{v \in A}\ q(D, A).$$

Then a pair (A_{opt}, v_{opt}) represents optimal attribute and a value, respectively, that should be chosen for a splitting.

6 Differentially private decision tree algorithm

After introducing decision tree algorithm, in this section we will see how one can achieve differentially private decision tree. Algorithm 2 presents a differential privacy adaptation of decision tree algorithm. The algorithm is very similar to Algorithm 1, so here we will focus on the parts of the algorithm concerning differential privacy.

The input parameters of the Algorithm 2 are the same as the input parameters of the Algorithm 1, except that the second algorithm has one additional parameter - a differential privacy budget denoted by ε. Here we will suppose that each sample in the dataset D pertains to one individual[5]. With N_{c_l} we will denote noisy count of elements in a subset D_{c_l}, and finally similar to D_c in non-private scenario, with $N_c = \{N_{c_l},\ l = 1, ..., k\}$ we will denote a set of noisy count of elements in subset D_{c_l}, if $N_{c_l} \neq 0$ (see Subsection 6.3 further ahead).

6.1 Differentially private mechanisms within decision tree algorithm

As we mentioned earlier, the noise that is introduced to a query depends on the sensitivity of the function applied to a dataset. In differentially private decision trees we will have two types of queries. One will be a counting query q_1, and the other one will be a splitting query q_2. Counting query is used in the part where the algorithm is calculating the number of appearances of each class in dataset D. Sensitivity of the counting query is 1, and for this evaluation Laplace mechanism is used. Splitting query request is conceptually more complex than the previous one, as it requires the use of the exponential mechanism.

The design of splitting query evaluation is shown in Figure 7. The exponential mechanism M_E takes as an input a dataset D, a differential privacy budget ε, and a quality function q, that can be gini index or max operator, as described in the preceding section. The mechanism returns the attribute and the value of that attribute that should be used later in branch creation. In this way, the answer of each splitting query request is fully differentially private, implying that the final model is also differentially private.

[5]This assumption is given in order to enable a reader better understanding of the algorithm. It is more intuitive to think about privacy preservation when a dataset refers to individuals.

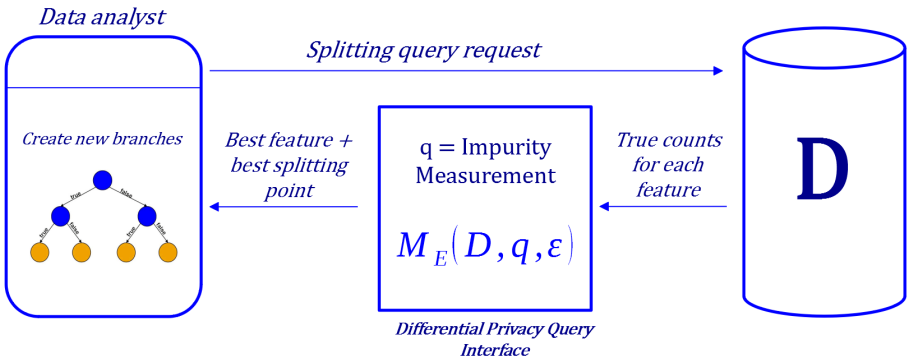

Figure 7: *Splitting query request*

6.2 Dividing a privacy budget

It is important to make a good plan about how privacy budget ε is going to be spent even before starting to create the model. Since decision tree works recursively, privacy budget should be divided through each recursion. One common way to divide the privacy budget is to split it in equal amounts to each level of the tree. Sequential and parallel composition properties help in here.

Figure 8 presents the idea of budget allocation within decision tree algorithm. Since by the construction of decision tree, the subsets that constitute each layer of the tree are disjoint (i.e., they have no common data points), parallel composition leads us to the conclusion that privacy budget should be equally split among different layers of the tree. From the root to the leaf we have $d + 1$ layers (including the root and the leaf), where d is the predefined tree depth. Applying the sequential composition it is obvious that each layer of the tree should get $\epsilon = \frac{\varepsilon}{d+1}$ amount of privacy budget[6].

[6]Suppose that we are given a tree with 2 layers, and suppose that we want to allocate privacy budget ε among those layers. Dataset D (first layer) and subsets D_1 and D_2 (second layer) have records in their intersection, which implies that a privacy budget ε should be equally spilt among them, i.e., D should get $\frac{\varepsilon}{2}$ and D_1 and D_2 should get $\frac{\varepsilon}{2}$ together. Subsets D_1 and D_2 (second layer) will be disjoint, therefore, there is no need to split the privacy budget among those two, i.e., each of them should get $\frac{\varepsilon}{2}$ amount of privacy budget (instead of getting $\frac{\varepsilon}{2(2)}$).

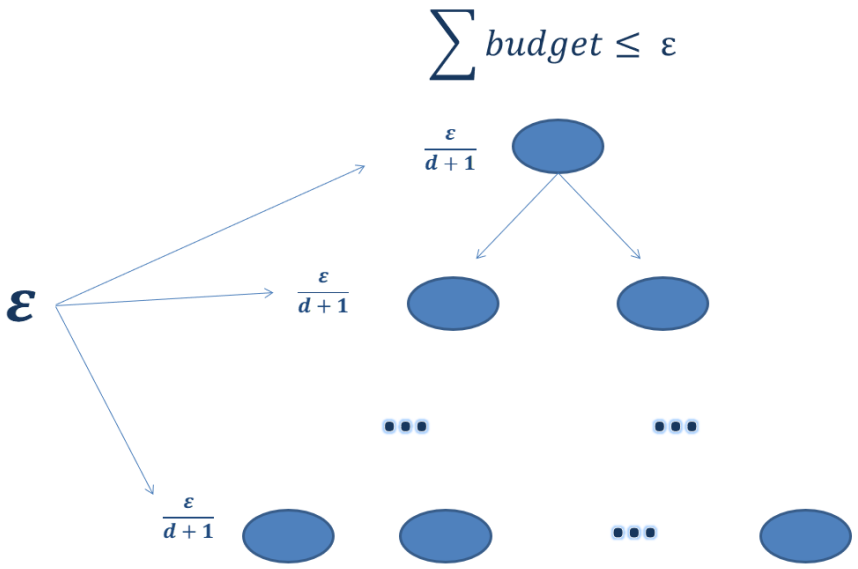

Figure 8: *Dividing a Privacy Budget*

As we mentioned earlier, in each node of the tree we will have two types of queries (qounting query q_1 and splitting query q_2). So we need to split privacy budget among those queries also. If each node gets $\frac{\varepsilon}{d+1}$ amount of privacy budget, then we can easily conclude that each query in the node should get $\frac{\varepsilon}{2(d+1)}$ amount of privacy budget. Finally, if we take into account that once a stopping criteria is satisfied, algorithm will not create any further branches (which implies that in leaf nodes only q_1 will be executed), we can conclude that each query should get $\frac{\varepsilon}{2(d+1)-1}$ amount of privacy budget. The idea of budget allocation is taken from [30].

6.3 Creating a private tree

In this subsection we will explain the workflow of the Algorithm 2.

Procedure `BuildDiffPDTree` is responsible for node creation, as it was the case with the procedure `BuildTree` from the Algorithm 1. In each execution, procedure first calculates query q_1 that returns the noisy count of elements in subset D_{c_l}. Laplace mechanism is applied on subsets D_{c_l} in order to calculate N_{c_l}. Subsets D_{c_l} are disjoint, i.e., $\bigcap_{l=1}^{k} D_{c_l} = \emptyset$, so we

can apply parallel composition to conclude that there is no need to split the privacy budget on l equal parts. After this, two additional concepts are derived based on the noisy counts N_{c_l} : N_c and N_n.

$N_c = \{N_{c_l},\ l = 1, ..., k\}$, represents a set that contains all N_{c_l}, such that $N_{c_l} \neq 0$ and $N_n = \sum_{l=1}^{k} N_{c_l}$ represents noisy counts of records in dataset D[7], Both of them will serve as a stopping criterion. If the stopping criteria are satisfied, the procedure chooses the maximum among calculated noisy class count N_{c_l} and returns as a leaf node labeled with a majority class. If convergence conditions are not satisfied, procedure searches for the best value for further splitting using exponential mechanism. Attributes with better scores with respect to the quality function q have higher probability of being chosen. After finding the best value for splitting, procedure splits the dataset based on the chosen attribute and continues the procedure iteratively on created subsets. The output of the algorithm is a differentially private decision tree model.

Please note that in differentially private decision tree we assume that the attribute schema is publicly available, including the description (domain, data type, etc.) of each of the attribute. In differentially private data mining, an analyst does not have an insight into the true dataset, but in order to be able to create a model, analyst must receive (from data owner) some basic information about the attributes.

[7]In order to save up the privacy budget, instead of calculating N_n directly with Laplace mechanism (in that scenario N_n would be calculated on the following way: $N_n = M_L(D, count(\cdot), \epsilon)$), we choose to calculate N_n by summing already calculated noisy number of records in subsets D_{c_l}

Algorithm 2 Differentially Private Decision Tree algorithm

1: **procedure** DIFFPDTREE($D, \mathcal{A}, C, d, q, m_s, \varepsilon$)
    Input: D - dataset, $\mathcal{A}$=$\{A_1,\ A_2,...,\ A_d\}$ - a set of attributes, C - class attribute, d - maximal tree depth, q - quality function, m_s - minimal sample threshold, ε - differential privacy budget
    $\epsilon = \frac{\varepsilon}{2 \cdot (d+1) - 1}$
    BuildDiffPDTree($D, \mathcal{A}, C, d, q, m_s, \epsilon$)
    return Differentially Private Decision Tree model
2: **end procedure**
3: **procedure** BUILDDIFFPDTREE($D, \mathcal{A}, C, d, q, m_s, \epsilon$)
    $N_{c_l} = M_L(D_{c_l}, count(\cdot), \epsilon),\ l = 1, ..., k$
    $N_c = \{N_{c_l},\ l = 1, ..., k\}$
    $N_n = \sum_{l=1}^{k} N_{c_l}$
4: **if** $\mathcal{A} = \emptyset$ or $d = 0$ or $|N_c| = 1$ or $N_n <= m_s$ **then**
    return a leaf labeled with $\text{argmax}_{c_l} N_{c_l}$
5: **end if**
    $(A_{opt}, v_{opt}) = M_E(D, q, \epsilon)$
    $(D_1, D_2) = Split(D, A_{opt})$
    BuildDiffPDTree($D_1, \mathcal{A} \setminus A_{opt}, C, d-1, q, m_s, \epsilon$)
    BuildDiffPDTree($D_2, \mathcal{A} \setminus A_{opt}, C, d-1, q, m_s, \epsilon$)
    return Decision node
6: **end procedure**

The following subsection will introduce the quality functions that we used in our work.

6.4 Quality functions

It is important to mention that both mechanisms are highly dependent on the choice of the quality function. That is because the noise depends on the sensitivity of the quality function: higher sensitivity of the function implies higher noise, which then implies lower accuracy of the decision tree classifier. Information gain has the highest sensitivity and because of that we decide to focus on the other two functions, i.e., gini index and max operator.

Let us first introduce some basic notations which will be used in quality functions introduction. Let D, $\mathcal{A}$, C, D_1 and D_2 be defined as in Algorithm 2. Furthermore, Let n_1 and n_2 denote, respectively, the number of samples in D_1 and D_2. Finally, denote by n_{1_l} and n_{2_l} the number of appearance of class c_l in subsets D_1 and D_2, respectively, that is, $n_{1_l} = |D_{1c_l}|$ and $n_{2_l} = |D_{2c_l}|$. We will now introduce max operator quality function.

6.4.1 Max operator

Among all quality functions, max operator is the most intuitive one. The goal of the max operator is to choose an attribute A_j from the set of attributes $\mathcal{A}$ and the value v_{j_i} of the attribute A_j which will have the highest class counts in partitions D_1 and D_2. For each value v_{j_i} the algorithm first splits D into D_1 and D_2, after what max operator is applied. Max operator finds the most frequent class in D_1 and the most frequent class in D_2 (classes do not have to be the same), sums those two numbers and chooses for the splitting an attribute A_j and a value v_{j_i} which have the highest sum of maximums. More formally, the max operator can be represented as:

$$q_{\max}(D, \mathcal{A}) = \max_l n_{1_l} + \max_l n_{2_l}.$$

The sensitivity of the max operator is 1. That is the reason why max operator is the best choice for the quality function in the case of differentially private decision trees. More details about quality functions can be found in [29].

6.4.2 Gini index

Gini index is impurity measurement used in CART algorithm [28]. Gini index tells how often a randomly chosen sample from the dataset D could be labeled incorrectly if it was labeled randomly according to the distribution of labels in D_1 and D_2. We did not use original gini index function in our implementation. We used approximation of the function developed in [29]. It is not such a rare case that the function needs to be modified in order to apply differential privacy within it. Again, this is because the sensitivity of a function greatly impacts the amount of noise that needs to be introduced. In order to reduce the added noise sometimes it is recommended to redefine a function if it is possible. We will now define (the variant of) gini index used in this paper. Approximation of the gini index function that we use is:

$$q_{\text{gini}}(D, \mathcal{A}) = -\left(n_1 * \left(1 - \sum_{l=1}^{k} \left(\frac{n_{1_l}}{n_1}\right)^2\right) + n_2 * \left(1 - \sum_{l=1}^{k} \left(\frac{n_{2_l}}{n_2}\right)^2\right)\right).$$

The goal is to find value v_{opt} in which our function q_{gini} will reach the maximum. The sensitivity of the approximated gini index function is 2.

6.5 Stopping criteria

In the case of differenctially private decision tree, algorithm stops if [34]:

- All attributes of the dataset are used for splitting ($\mathcal{A} = \emptyset$);
- Number of iterations have achieved specified limit ($d = 0$).
- Noisy number of samples in the training set is less than a specified threshold ($N_n <= m_s$);
- There are instances representing only one class in the training set ($|N_c| = 1$).

6.6 Implementation in Python

We based implementations of differentially private decision trees on the implementation presented in [27]. The basic concept of the implementation of decision tree algorithm is the same as it is there, except the fact that the quality function used in [27] is Information gain. In this thesis we focus on max operator and gini index quality functions because they are less sensitive to noise. Primarily, we implemented a differentially private decision tree with the Information gain quality function, but since the accuracy score results of the classifier were not satisfying at all, we decided to give up this approach. We will now introduce our two implementations, with explanation concerning differential privacy.

Max operator implementation

Method `.fit ()` from the class `DecisionTree ()` builds a differentially private decision tree. The method takes as an input training set and total amount of privacy budget denoted by $EPSILON$. The method first divides the privacy budget according to the previously defined rules, after which the model is generated with function `build_tree`. Function `build_tree` takes privacy budget as a parameter, i.e., whenever the function is executed, $\frac{\varepsilon}{d+1}$ amount of budget is spent. This fits perfectly with a theoretical approach of budget allocation. Every execution of the function implies a creation of new node, which implies that each note gets $\frac{\varepsilon}{d+1}$ of privacy budget. Differential privacy mechanisms are applied in functions `Best_split` and `class_count`. As it was explained in subsection 6.2 (*Dividing a privacy budget*), each mechanism will get $\frac{\varepsilon}{2(d+1)-1}$ amount of privacy budget. Function `Best_split` calls function `max_operator` which returns the best value for splitting based on the max operator quality function. Exponential mechanism is used here to choose the best attribute for splitting the training set. Attributes with better rates (higher max operator value) are going to have higher probability of being chosen. Function `class_count` returns noisy class counts of the input dataset, which are then used for stopping criteria and for creation of the leaf nodes if the criteria are satisfied. Laplace mechanism is used for this evaluation.

```
import os
import numpy as np
import pandas as pd
from random import sample
from sklearn.metrics import accuracy_score
from numpy import linalg as LA

class DecisionTree (object):
    ''
    Decision Tree Algorithm
    param max_depth: maximum alowed depth of the tree
    param min_sample: minimum number of points in dataset
        in
    order to split it any further
    ''
    def __init__ (self, max_depth =5 , min_sample = 5,
   tree = None, columns = {}:
        self.max_depth = max_depth
        self.min_sample = min_sample
        self.tree = tree
        self.columns = columns

    def fit (self, data, EPSILON):
        epsilon=EPSILON/(2*(self.max_depth+1)-1)
        att_type = []
        for i in range (data.shape[1]-1):
26          if len (np.unique (data[:,i]))>20:
                att_type.append ('num')
            else:
                att_type.append ('cat')
        tree = build_tree (data, self.max_depth,
  self.min_sample, np.array (att_type), epsilon)
        self.tree = tree
        return tree

    def predict_proba (self, data):
        tree = self.tree
        if len (data.shape) == 1:
            return proba2 (proba1 (data, tree))
        if len (data.shape) == 2:
            res = [proba2 (proba1 (data[i], tree)) for i in
     range (len (data))]
            return res
```

```
    def predict (self, data):
        tree = self.tree
        if len (data.shape) == 1:
            return pred1 (data, tree)
        if len (data.shape) == 2:
            res = [pred1 (data[i], tree) for i in
   range (len (data))]
            return res

class Question:
    """
    A Question is used to partition a dataset.
    """

    def __init__ (self, column, value, att_type):
        self.column = column
        self.value = value
        self.att_type = att_type

    def match (self, data):
        if len (data.shape) == 1:
            val = data[self.column]
        if len (data.shape) == 2:
            val = data[:, self.column]
        if self.att_type[self.column] == 'num':
            return val >= self.value
        else:
            return val == self.value

def split (data, column, value, att_type):
    question  = Question (column, value, att_type)
    return data[question.match (data)],
   data[np.invert (question.match (data))]

def max_operator_value (Data, Att, value, att_type):

    X, Y = split (Data, Att, value, att_type)

    if float (len (X)) == 0 or float (len (Y)) == 0:
        return 0
    unique = np.unique (Data[:,-1])
    #labels of data points
    arrX = np.array (X[:,-1]) #class label of X
    arrY = np.array (Y[:,-1]) #class label of Y
```

```
    countX = np.array ([ (arrX == i).sum () for i in unique
        ])
    #count classes for value v in X
    max_X=max (countX) #choosing max class in X
    countY = np.array ([ (arrY == i).sum () for i in unique
        ])
    #count classes for value v in Y
    max_Y=max (countY)
    max_count=max_X+max_Y #sums those 2 maximums
    return max_count

def max_operator (Data, Att, att_type):
#applying max operator value to each value of attribute Att

    V = np.unique (Data[:,Att])
    if att_type[Att] == 'cat':
        g = [ (value, max_operator_value (Data, Att, value,
  att_type)) for value in V]
    else:
        g = [ (value, max_operator_value (Data, Att, value,
  att_type)) for value in V]
    val, max_score = max (g, key=lambda t: t[1])
   #max_score returns sum of maximums of partitions

    return val, max_score


#exponential mechanism for max operator
def Best_split (data, att_type, epsilon):
    val=[]
    evals = []
    const=epsilon/2 #sensitivty of max operator is 1
    for Att in range (data.shape[1]-1):
        val.append (max_operator (data, Att, att_type)[0])
        evals.append (max_operator (data, Att, att_type)
            [1])
    weights=np.exp (const*np.array (evals))
    prob=weights/LA.norm (weights, 1)
    max_choise = np.random.choice (np.array (evals), p=prob
        )

    att=evals.index (max_choise)
    val=val[evals.index (max_choise)]

    return att, val
```

```
130 def class_count (a, epsilon):
    unique = np.unique (a[:,-1])
    arr = np.array (a[:,-1])
    count = np.array ([np.random.laplace ( (arr == i).sum
        (),
  1/epsilon) for i in unique])
    dic=dict (zip (unique, count))
    max_class=max ([ (k,v) for k,v in dic.items ()],
   key = lambda t: t[1])[0]
    return dic, max_class

class Leaf:
    def __init__ (self, rows, epsilon):
        self.predictions = class_count (rows, epsilon)[0]
  # for predicting probabilities
        self.category = class_count (rows, epsilon)[1]
  # for predicting class

class Decision_Node:
    def __init__ (self, left_branch, right_branch, att, val
        ,
    itter, question):
        self.left_branch = left_branch
        self.right_branch = right_branch
        self.att = att
        self.val = val
        self.itter = itter
        self.question = question
156
def build_tree (data, max_depth, min_sample,att_type,
    epsilon
    classes ,itter = 0):
    att_type = att_type
    classes = Leaf(data, epsilon)
    a=classes.predictions
    if  itter == max_depth or data.shape[1]-1==0 or
       sum(a.values()) <= min_sample or len(a) == 1:

        return classes
    att, val = Best_split (data, att_type, epsilon)

    question = Question (att, val, att_type)

    left_data, right_data = split (data, att, val, att_type
```

```
        )
    itter += 1

    if att_type[att] =='cat':
        left_data = np.delete (left_data, att, 1)
        right_data = np.delete (right_data, att, 1)
        att_type = np.delete (att_type, att)

    left_branch = build_tree (left_data, max_depth,
 min_sample, att_type, epsilon, itter)
    right_branch = build_tree (right_data, max_depth,
 min_sample, att_type, epsilon, itter)

    return Decision_Node (left_branch, right_branch, att,
        val
    ,itter, question)

def proba1 (row, node):
    '''
    counts occurance of classes in each leaf
    '''
    # Base case: we've reached a leaf
    if isinstance (node, Leaf):
        return node.predictions
    # Decide whether to follow the left or right branch.
    if node.question.match (row):
        return proba1 (row, node.left_branch)
    else:
        return proba1 (row, node.right_branch)

def proba2 (counts):
    '''
    presents counts in percents
    '''
    total = sum (counts.values ()) * 1.0
    probs = {}
    for lbl in counts.keys ():
       probs[lbl] = round (counts[lbl] / total, 2)
    return probs

def pred1 (row, node):
    '''
    picks majority class in each leaf
    '''
    if isinstance (node, Leaf):
```

```
        return node.category
    if node.question.match(row):
        return pred1(row, node.left_branch)
    else:
        return pred1(row, node.right_branch)
```

Listing 1: *Python Implementation of Differentially Private Algorithm with Max Operator quality function*

Gini index implementation

Implementation of the differentially private decision tree with the gini index quality function is similar to the previous one, we only used gini index instead of max operator quality function. So, by replacing part of the previous code starting from line 78 all the way to line 130, with the code that follows next, we will get a complete implementation of the differentially private decision tree with the gini index quality function.

```
def gini_ent(X):
    E = 0
    n = len(X)
    for i in np.unique(X):
        m = len(X[X == i])
        if m != 0:
            E += np.exp2(m/n)
    return 1-E

def gini_index_value(Data, Att, value, att_type):
    X, Y = split(Data, Att, value, att_type)
    if float(len(X)) == 0 or float(len(Y) == 0):
        return 0.0
    gini = -(len(X)*gini_ent(X[:,-1])+len(Y)*
     gini_ent(Y[:,-1]))
    return gini

def gini_index(Data, Att, att_type):
    V = np.unique(Data[:,Att])
    if att_type[Att] == 'cat':
        g = [(value, gini_index_value(Data, Att, value,
      att_type)) for value in V]
```

```
    else:
        g = [ (value, gini_index_value (Data, Att, value,
      att_type)) for value in V]
    val, gini = max (g, key=lambda t: t[1])
    return val, gini

#exponential mechanism for gini index
def Best_split (data, att_type, epsilon):
    val=[]
    evals = []
    const=epsilon/4 #sensitivty of gini index is 2
    for Att in range (data.shape[1]-1):
        val.append (gini_index (data, Att, att_type)[0])
 #value for attribut
        evals.append (gini_index (data, Att, att_type)[1])
 #gini index for that value
    weights=np.exp (const*np.array (evals))
    prob=weights/LA.norm (weights, 1)
    max_choise = np.random.choice (np.array (evals), p=prob
    )
    att=evals.index (max_choise)
    val=val[evals.index (max_choise)]
    return att, val
```

Listing 2: *Python Implementation of Differentially Private Algorithm with Gini Index quality function*

The output of the algorithms are models that ensure differential privacy, i.e., the training dataset is fully protected from anyone. This means that not a single information about the individual records in the datasets can be learned from the final model. More details can be found in [30]. The idea for the implementation of differentially private decision tree is taken from [29].

It should be emphasized that in differential privacy schemes presented in this thesis, the privacy of individuals in training data is protected, but the privacy of individuals in testing data is not. The privacy of individuals in the training data must be protected, because data analyst is involved in the feature and model design (model training), while in the prediction stage, data analyst does not need to be involved in the prediction process. The model can be deployed on a security guaranteed platform and automatically runs. In this case, privacy protection is not necessary [30].

7 Experimental results

In this chapter, we introduce the datasets that were used for testing described Algorithms and discuss the classification results. We compare our results of differentially private decision trees with the results of non-private decision tree developed in [27].

7.1 Experimental setup

We tested our algorithms on a single laptop computer with the following specifications:

- processor: Intel® Core^{TM} i7-8750H (2.2 GHz base frequency, up to 4.1 GHz with Intel® Turbo Boost Technology, 9 MB cache, 6 cores),
- installed memory (RAM): 16 GB DDR4-2666 SDRAM (2 x 8 GB),
- system type: 64-bit Operating System, x64-based processor.

7.2 Programming language

Algorithms were implemented and tested in Python programming language [43], [26].
Nowadays Python is widely used across multiple domains in computer science, especially in data science. This is because it is known to be an intuitive language, it is open source and is easy to work with. Another reason why it is so popular in data science community is because of its wide range of libraries which are suitable for machine learning and statistical analysis. Some of the libraries which were helpful to us through our work are numpy [38], pandas [39], scipy [42], matplotlib [41], seaborn [40], scikit-learn [45] etc.
In our work we used Python 3.6 version and we wrote the codes in Spyder environment [44].

7.3 Metrics, cross–validation and privacy budget

Accuracy score

Since we focused on the classification problem, for measuring the utility of our algorithms we used accuracy score.
Scikit Learn library has built in function `accuracy_score` that takes as an input predicted and true values and calculates the ratio of correctly predicted instances. The range of the function is [0, 1]. If the accuracy score takes value 0, that means that all predicted labels are wrong, and if it takes value 1, that means that they are all correct.

Cross–validation and privacy budget

Cross–validation is a machine learning technique that is used as a tool to avoid overfitting. Overfitting means that the model we are fitting is too much adapted to the training data (model is giving perfect results on the training data, but not so good results on the test data). *K-fold* cross–validation is one of the most widely used cross–validation techniques. It splits dataset into k different subsets. $K-1$ subsets are used for training the model, and the remaining one is used for testing. Algorithm fits the model k times. At the end it returns as a result the average accuracy value.
In our experiments, we executed 5 runs of 5-fold cross–validation. For each run we evaluated the results for seven different values of privacy budget, i.e., ε takes values from the following subset: $\{0.5, 0.75, 1, 2, 3, 4, 5\}$. Although the value of privacy budget should be relatively small (less than 1), in our experiment we used larger values because datasets used for evaluation are not so big.

7.4 Datasets

In this thesis we focus on categorical data. All datasets were downloaded from the UCI machine Learning Repository [45]. We will now briefly describe each of them.

Amphetamine, Benzodiazepine, Cocaine, Ecstasy, Legal highs, Magic Mushrooms and Nicotine dataset

All seven datasets are derived from a Drug consumption dataset. Dataset contains 1885 records. In this dataset there are 12 features describing each respondent. Features represent personality measurements which include

NEO-FFI-R (neuroticism, extraversion, openness to experience, agreeableness, and conscientiousness), BIS-11 (impulsivity), and ImpSS (sensation seeking), level of education, age, gender, country of residence and ethnicity. Participants were questioned about their usage of 18 legal and illegal drugs (alcohol, amphetamines, amyl nitrite, benzodiazepine, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, mushrooms, nicotine and volatile substance abuse and one fictitious drug (Semeron)). Each label variable contains seven classes: "Never Used", "Used over a Decade Ago", "Used in Last Decade", "Used in Last Year", "Used in Last Month", "Used in Last Week", and "Used in Last Day".
For testing our algorithms we made seven datasets, all containing the same features and different label variables[8]. This dataset can be used for multiclass classification problems or it can be transformed into binary classification problem. We choose the second approach. We made two new classes by union of part of classes in the following way: "Never Used", "Used over a Decade Ago" form class "Non-user" and all other classes form class "User".

Mushroom, Car Evaluation and Tic-Tac-Toe dataset
Again, all three datasets are publicly available at UCI Machine Learning repository. Here we almost did not have to do any data preprocessing work, i.e., datasets were "clean" and ready to use.
Mushroom dataset This dataset consists of 8124 instances with 22 attributes and this dataset is used for binary classification problem.
Car Evaluation dataset This dataset consists of 1728 instances. Number of attributes in Car Evaluation dataset is 6. Each record can take one of the 4 possible classes. This is the only dataset which is used for multi class classification.
Tic-Tac-Toe dataset In this dataset, there are 958 incstances described with 9 features. All instances can take one of the two possible classes.

[8]We choose 7 (*Amphetamine, Benzodiazepine, Cocaine, Ecstasy, Legal highs, Magic Mushrooms and Nicotine*) of 18 drugs, and extracted 7 new datasets where each of them is related to one drug. Features in the datasets are always the same namely: *neuroticism, extraversion, openness to experience, agreeableness, conscientiousness, impulsivity, sensation seeking, level of education, age, gender, country of residence and ethnicity* and datasets differ in their class variables. For example, in case of Cocaine dataset, class variable is describing individual's usage of Cocaine, in case of Ecstasy dataset, class variable is describing individual's usage of Ecstasy, etc.

7.5 Results

Obtained results are shown in figures and tables below.
We can see that, in most cases, decision tree with the max operator quality function performs better than the same classifier with the gini index quality function. The reason for this is the fact that the sensitivity of the quality function influences the magnitude of noise introduced to the exponential mechanism. In other words, for the same privacy parameter ε, exponential mechanism will have different effectiveness for different quality functions [29]. So, decision tree with the max operator quality function performs the best because max operator is the least sensitive to noise.
In theory, for larger values of privacy budget, classifiers should have better accuracy score results. In our experiments this was not always the case. The reason for this is the small size of training set. Because datasets were relatively small, Laplacian noise added to a class counts in a leaf node has more prominent effect on the accuracy score results.
A drawback of our algorithms is high variance of the experimental results. The introduced noise cannot be controlled and is not the same in each evaluation, i.e., introduced randomness leads to different results in each algorithm. Also, for the same reason curves describing accuracy score are not strictly increasing as they should be theoretically. The solution to this problem can be higher number of folds in cross-validation process. Since the accuracy score results of the private classifiers depend on the noise introduced by differential privacy, by increasing the parametar K in cross-validation technique we will get more stable results, since the introduced noise will be sampled over the larger number of iterations.
Another reason for not getting strictly increasing results is high dimensionality of a dataset. For example, accuracy of the Mushroom dataset tends to decrease more comparing to other datasets. This is because the dimensionality (as measured by the number of features) of the dataset is 22. In summary, we got to the conclusion that small dataset size and high dimensionality affect the results of private classification much more than they do in non-private scenario.
Note that the quality function used in non-private classifier is Information gain, while the quality funcitons used in private classifiers are max operator and gini index. For that reason in some cases it can happen that accuracy score results of the private classifiers can reach higher values than the accuracy score results of non-private classifier.

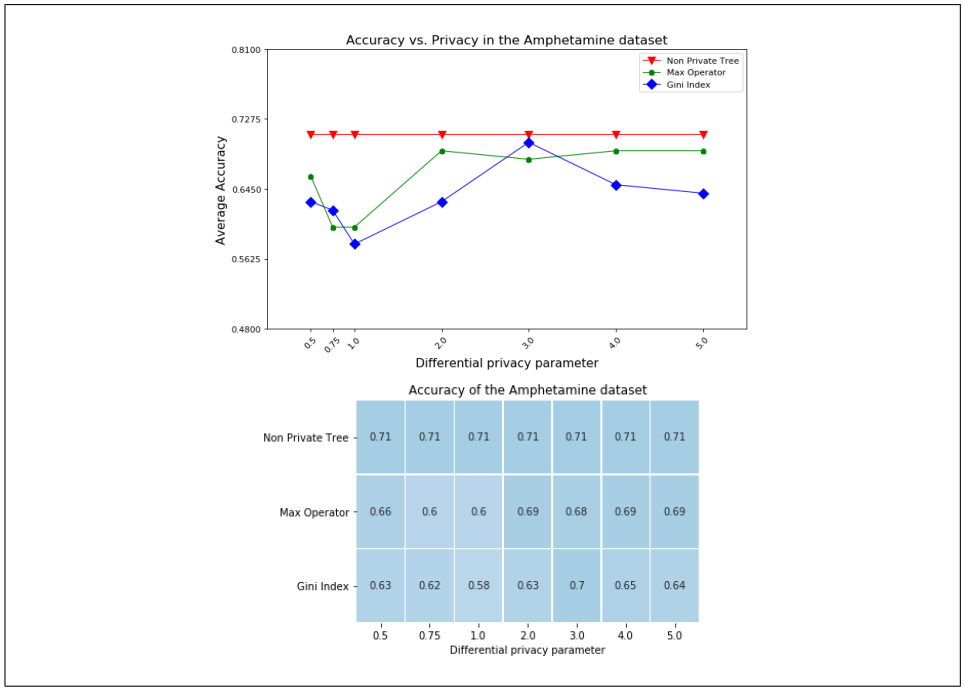

Figure 9: *Performance on Amphetamine dataset*

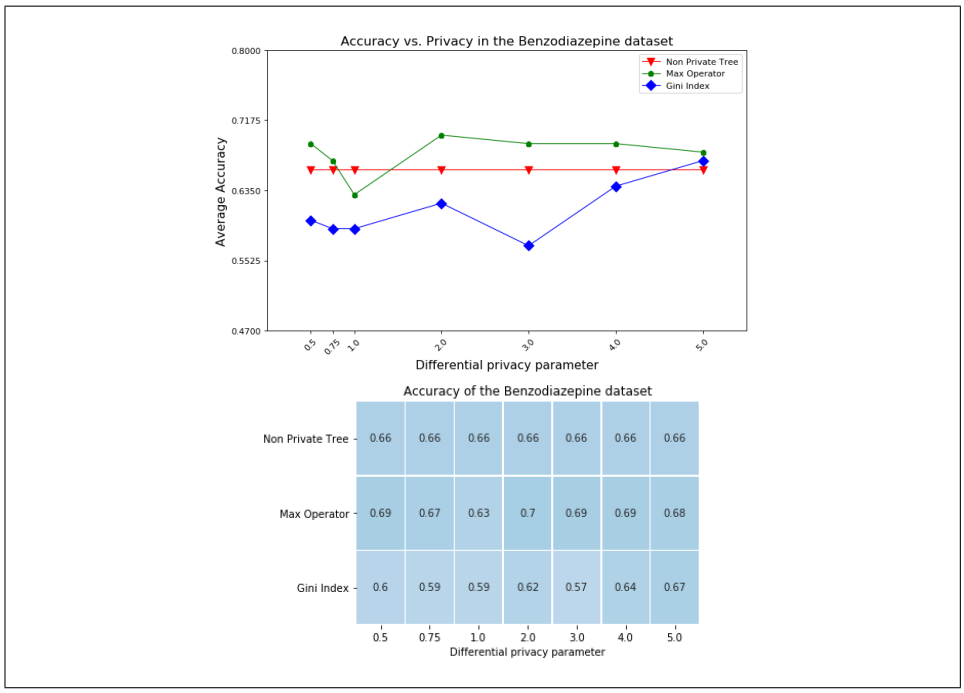

Figure 10: *Performance on Benzodiazepine dataset*

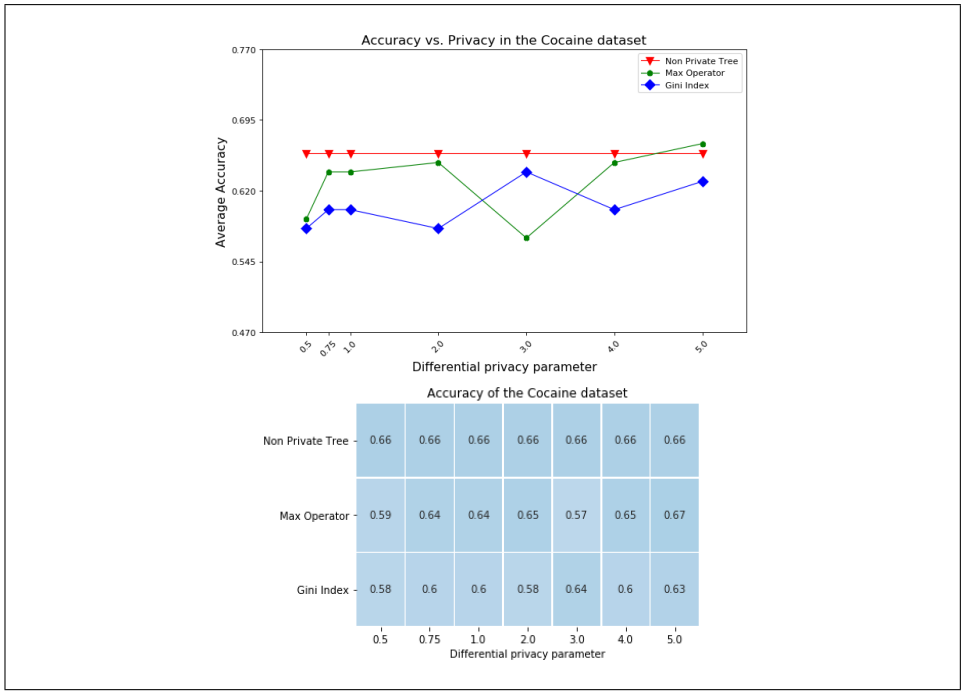

Figure 11: *Performance on Cocaine dataset*

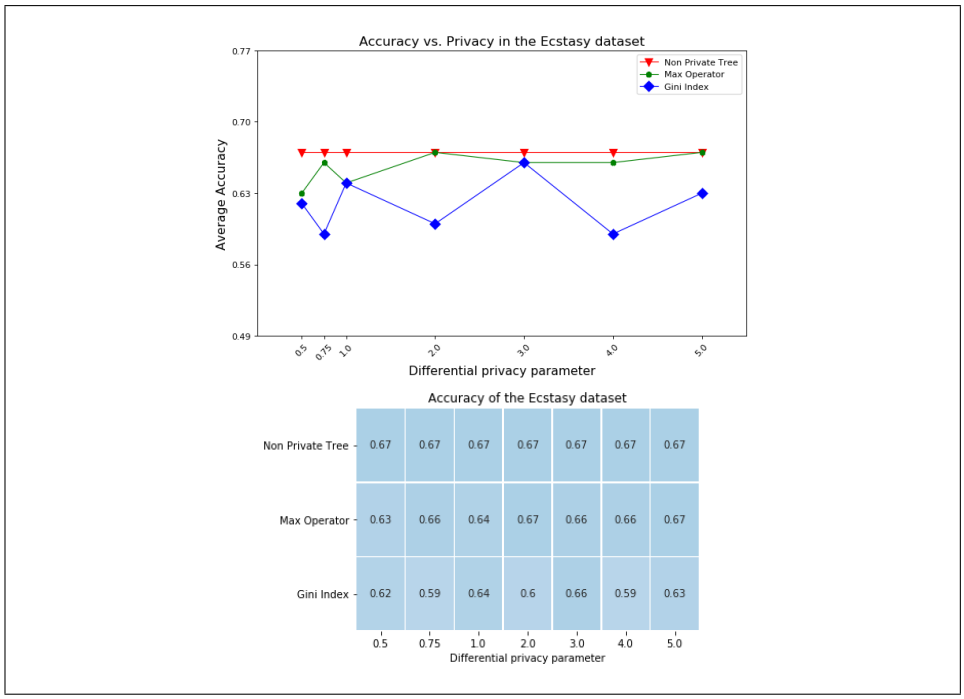

Figure 12: *Performance on Ecstasy dataset*

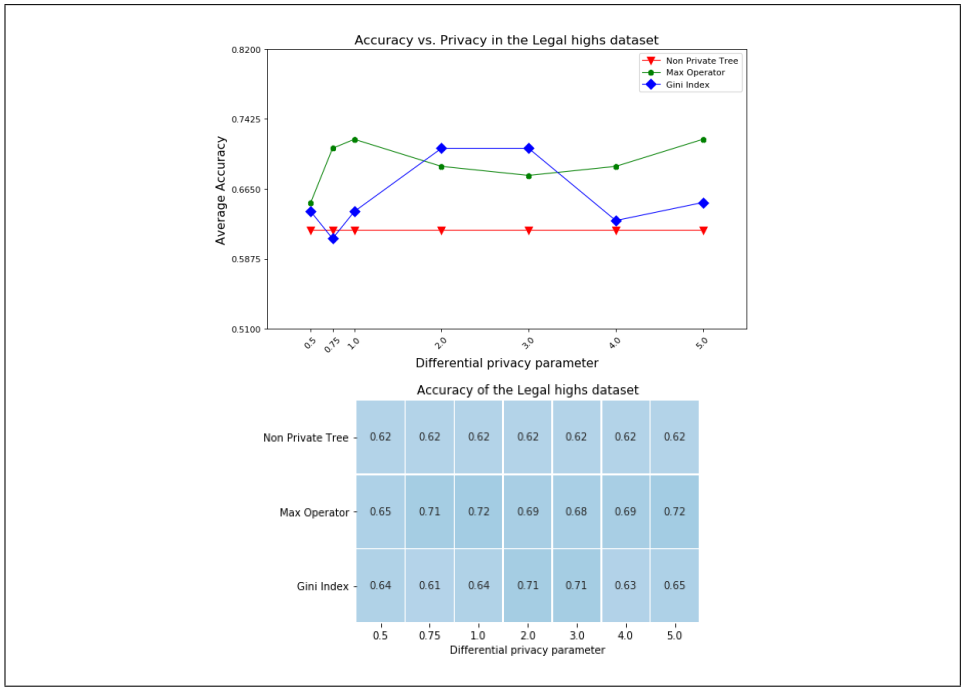

Figure 13: *Performance on Legal highs dataset*

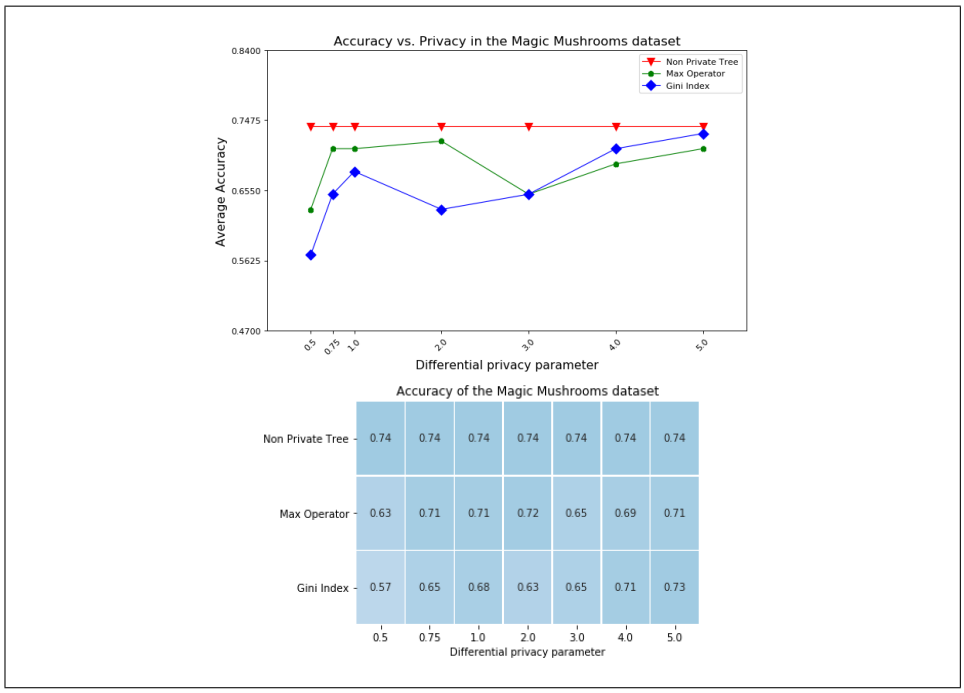

Figure 14: *Performance on Magic Mushrooms dataset*

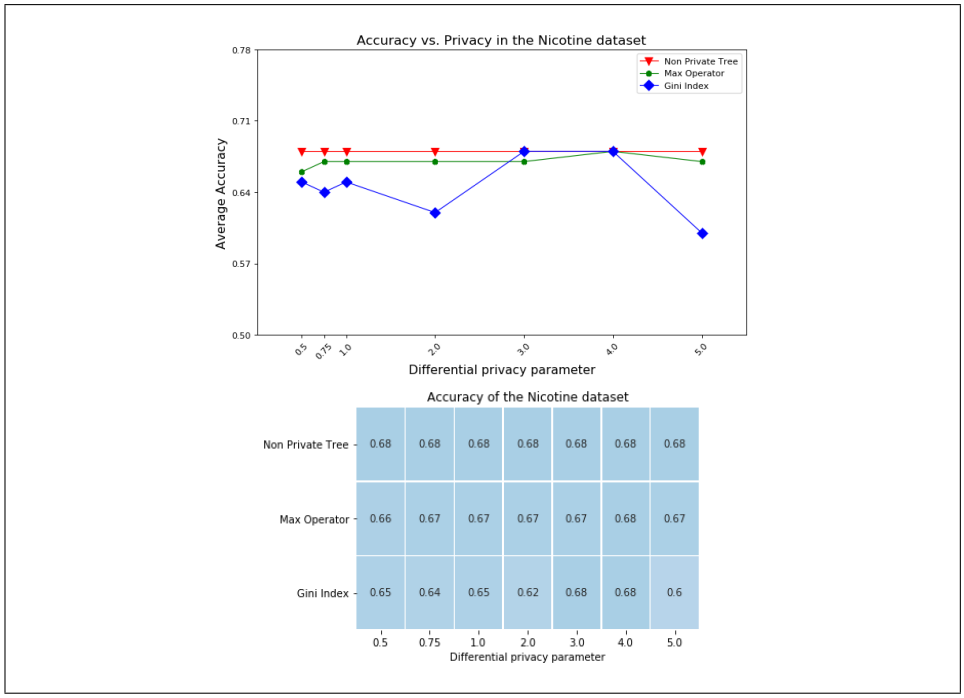

Figure 15: *Performance on Nicotine dataset*

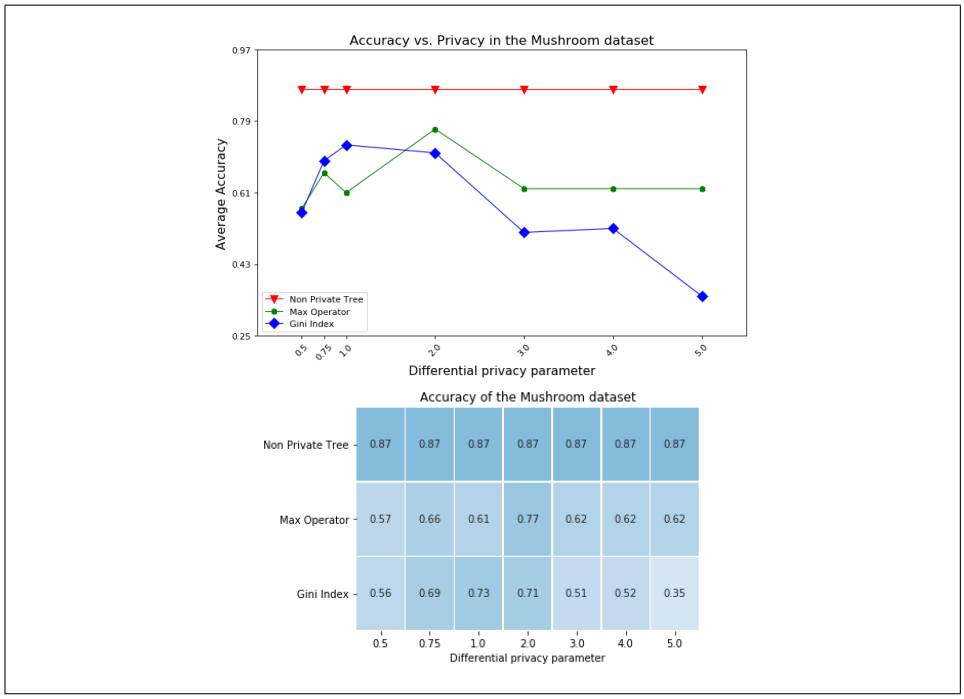

Figure 16: *Performance on Mushroom dataset*

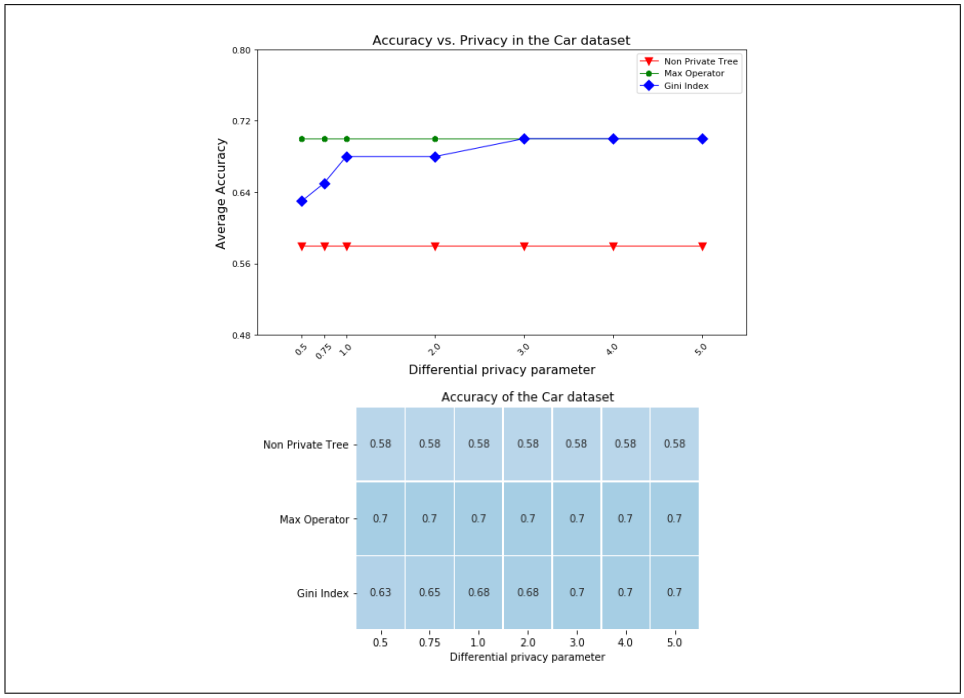

Figure 17: *Performance on Car Evaluation dataset*

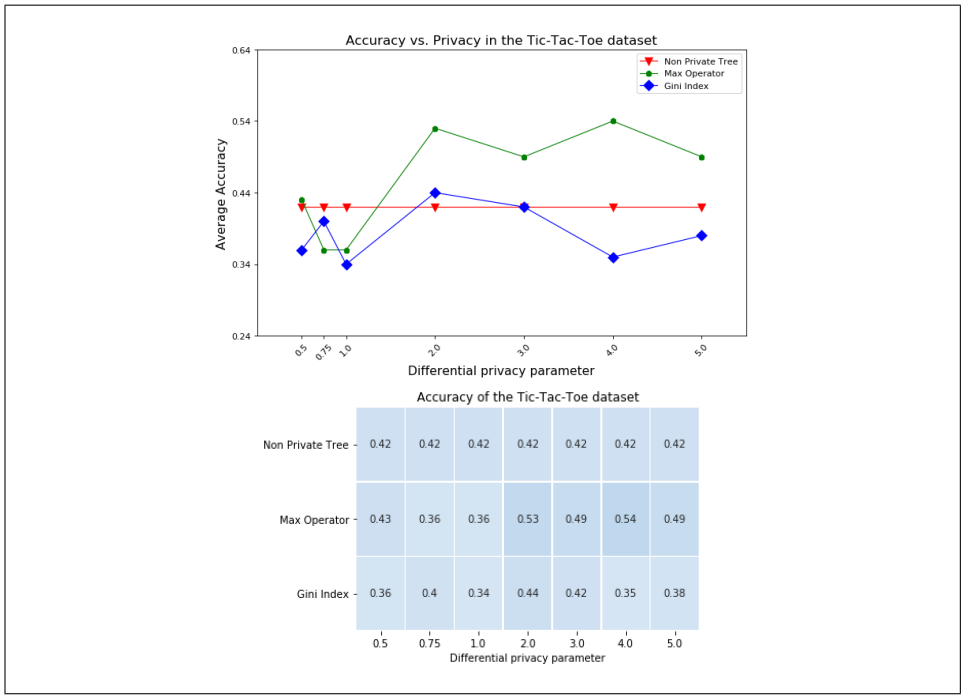

Figure 18: *Performance on Tic-Tac-Toe dataset*

8 Conclusion

In this work, we described two ways to build differentially private decision tree classifier. For both algorithms we presented the theoretical background, the implementation in Python and the performance on several real public datasets.
We illustrated that it is possible to achieve good prediction accuracy while preserving the privacy of the individual records in the datasets.

The final conclusion is that in order to achieve differential privacy, sometimes machine learning models need to be modified. The approaches which are giving the best results in non-private scenario may not be giving the best results in the case of differentially private models. As we saw in our experiments, the sensitivity of the queries becomes crucial to performance of the algorithm when differential privacy is involved. Since we are planning to improve our work, some of the future challenges will include:

- finding a solution to reduce the variance in the experimental results;
- handling numerical features;
- exploring different strategies for choosing/optimizing ε;
- implementing differentially private random forest.

Differential privacy is gaining a significant foothold in data mining and machine learning, evident by the recent surge in papers concerning the application of differential privacy [30], [31], [32], [35]. Different architectures and approaches for building differentially private decision trees can be found in [33].

9 Appendix

In this additional section we presented script developed for testing our differentially private classifiers.
Script contains two functions `test` and `plot_results`. Function `test` performs K-fold cross–validation, trains the model on a given dataset and, finally, calculates and returns prediction accuracy of a given classifier. Function `plot_results` is used for visual representation of classification results.

Testing script

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score

from Max_Operator import DecisionTree as MaxTree
from Gini_Index import DecisionTree as GiniTree
from DT import DecisionTree as BasicTree

MaxOperator=MaxTree ()
GiniIndex=GiniTree ()
NonPrivateTree=BasicTree ()


'''Function test is used for trainig and testing
    classifiers
 (cross validation, measuring accuracy, averaging the
    results)
  Function plot_results is used for visual representation
of classification results.
'''

def test (data, name_of_data, criterion_function,
    Kfold_splits, epsilon):
    X=data[:, :-1]
    y=data[:, -1]
    kf = KFold (n_splits=Kfold_splits)
    accuracy=0
    for train, test in kf.split (X, y):
```

```
        if criterion_function==NonPrivateTree:
            criterion_function.fit (data[train])
        else:
            criterion_function.fit (data[train], epsilon)
        res=criterion_function.predict (X[test])
        accuracy+=accuracy_score (res, y[test])/
    Kfold_splits

    return round (accuracy, 2)

def plot_results (matrix_results, epsilon, Crit_functions,
 Crit_functions_names, name_of_data, itteration):
    plt.figure (figsize= (10, 6), dpi=80)
    plt.subplot (1, 1, 1)
    colors=['red','green','blue']
    markers=['v','*','D']
    for i in range (matrix_results.shape[0]):
        plt.plot (epsilon, matrix_results[i], color=colors[
    i]
  ,markeredgewidth=3.0, marker=markers[i],
  linewidth=1.0, linestyle="-",
  label=Crit_functions_names[i])
    plt.xlim (min (epsilon)-0.5, max (epsilon)+0.5)
    plt.xticks (epsilon, epsilon, fontsize=10, rotation=45)
    plt.ylim (round (matrix_results.min ()-0.1, 2),
 round (0.1+matrix_results.max (), 2))
    plt.yticks (np.linspace (round (matrix_results.min ()
    -0.1,
  2), round (0.1+matrix_results.max (), 2), 5,
  endpoint=True))
    plt.legend (loc='best')
    plt.xlabel ('Privacy Budget', fontsize=10)
    plt.ylabel ('Average Accuracy', fontsize=10)
    plt.title ('Accuracy vs. Privacy in the '+name_of_data+
  ' dataset', fontsize=15)
    plt.savefig ('Accuracy vs. Privacy in the '+
    name_of_data+
   'dataset.png', format='png', dpi=330,bbox_inches='tight'
    )
    plt.show ()


    hmap=pd.DataFrame (data=matrix_results,
  index = Crit_functions_names, columns=epsilon)
    ax=sns.heatmap (hmap, vmin=0, vmax=2, annot=True,
```

```
 linewidths=.5, cbar=False, cmap='Blues')
   ax.set_title ('Accuracy of the '+name_of_data+' dataset
   ')
   ax.set_xlabel ('Differential privacy parameter',
  fontsize=10)
   plt.savefig ('Accuracy of the '+name_of_data+' dataset
.png', format='png', dpi=330, bbox_inches='tight')
   plt.show ()

   return
```

Listing 3: *Testing Private Classifiers*

10 List of Figures

1 Definition of Differential Privacy

2 Laplace distribution

3 Simplified scheme of Interactive approach

4 Simplified scheme of Non-Interactive approach

5 Central Differential Privacy

6 Local Differential Privacy

7 Splitting query request

8 Dividing a Privacy Budget

9 Performance on Amphetamine dataset

10 Performance on Benzodiazepine dataset

11 Performance on Cocaine dataset

12 Performance on Ecstasy dataset

13 Performance on Legal highs dataset

14 Performance on Magic Mashrooms dataset

15 Performance on Nicotine dataset

16 Performance on Mushroom dataset

17 Performance on Car Evaluation dataset

18 Performance on Tic-Tac-Toe dataset

11 List of Listings

1 Implementation of Differentially Private Algorithm with Gini Index quality function

2 Implementation of Differentially Private Algorithm with Max Operator quality function

3 Testing Differentially Private Algorithms

12 List of Algorithms

1 Decision Tree Algorithm

2 Differentially Private Decision Tree Algorithm

References

[1] L. Sweeney. *Achieving k-anonymity privacy protection using generalization and suppression*. Int. J. Uncertain. Fuzziness Knowl.-Based Syst., 10 (5):571–588, Oct. 2002.

[2] Daniel C. Barth-Jones. *The 'Re-Identification' of Governor William Weld's Medical Information: A Critical Re-Examination of Health Data Identification Risks and Privacy Protections, Then and Now*. Tech. rep. Columbia University - Mailman School of Public Health, Department of Epidemiology, July 2012.

[3] J. Domingo-Ferrer and V. Torra. *A Critique of k-Anonymity and Some of Its Enhancements*. In Third International Conference on Availability, Reliability and Security (ARES 08), pages 990–993. IEEE, 2008.

[4] Cynthia Dwork and Aaron Roth. 2014. *The Algorithmic Foundations of Differential Privacy*. Found. Trends Theor. Comput. Sci. 9, 3–4 (August 2014), 211-407. DOI=http://dx.doi.org/10.1561/0400000042.

[5] Cynthia Dwork. *An Ad Omnia Approach to Defining and Achieving Private Data Analysis*. In: Proceedings of the 1st ACM SIGKDD International Conference on Privacy, Security, and Trust in KDD. PinKDD'07. San Jose, CA, USA: Springer-Verlag, 2008, pp. 1–13. ISBN: 3-540-78477-2, 978-3-540-78477-7.

[6] Cynthia Dwork. *Differential Privacy*. In: Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II. Ed. by Michele Bugliesi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 978-3-540-35908-1.

[7] Cynthia Dwork. *Differential Privacy: A Survey of Results*. In: Proceedings of the 5th International Conference on Theory and Applications of Models of Computation. TAMC'08. Xi'an, China: Springer-Verlag, 2008, pp. 1–19. ISBN: 3-540- 79227-9, 978-3-540-79227-7.

[8] Cynthia Dwork et al. *Calibrating Noise to Sensitivity in Private Data Analysis*. In: Theory of Cryptography: Third Theory of Cryptography

Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32732-5.

[9] Moritz Hardt and Kunal Talwar. *On the Geometry of Differential Privacy*. In: CoRR abs/0907.3754 (2009).

[10] Chao Li and Gerome Miklau. *An Adaptive Mechanism for Accurate Query Answering under Differential Privacy*. In: CoRR abs/1202.3807 (2012).

[11] Frank McSherry. *Privacy Integrated Queries: An Extensible Platform for Privacy-preserving Data Analysis*. In: Commun.ACM53.9 (Sept. 2010), pp. 89– 97. ISSN: 0001-0782.

[12] Frank McSherry and Kunal Talwar. *Mechanism Design via Differential Privacy*. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science. FOCS '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 94–103. ISBN: 0-7695-3010-9.

[13] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. *Smooth Sensitivity and Sampling in Private Data Analysis*. In: Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing. STOC '07. San Diego, California, USA: ACM, 2007, pp. 75–84. ISBN: 978-1-59593-631-8.

[14] V. N. Vapnik and A. Ya. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*. In: Theory of Probability and its Applications 16.2 (1971), pp. 264–280.

[15] Tianqing Zhu et al. *Differential Privacy and Applications*. Springer International Publishing AG 2017. ISBN 978-3-319-62004-6.

[16] *Learning wirh Privacy at Scale*. Differential Privacy Team. Apple. https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf

[17] *https://www.apple.com/ios/health/*

[18] *RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response*. Úlfar Erlingsson and Vasyl Pihur and Aleksandra Korolova. 2014, https://arxiv.org/abs/1407.6981. Proceedings of the 21st ACM Conference on Computer and Communications Security, Scottsdale, Arizona.

[19] Duy Vu and Aleksandra Slavkovic. 2009. *Differential Privacy for Clinical Trial Data: Preliminary Evaluations.* In IEEE International Conference on Data Mining Workshops. IEEE, 138–143.

[20] J. Lee and C. Clifton. *How much is enough? choosing ε for differential privacy.* International Conference on Information Security. Springer, 2011, pp. 325–340.

[21] X.-M. He, X. S. Wang, H.-H. Chen, and Y.-H. Dong, *Study on choosing the parameter ε in differential privacy* Journal on Communications, vol. 36, no. 12, pp. 124–130, 2015.

[22] J. Hsu, M. Gaboardi, A. Haeberlen, S. Khanna, A. Narayan, B. C. Pierce, and A. Roth, *Differential privacy: An economic method for choosing epsilon.* Computer Security Foundations Symposium (CSF), IEEE 27th. IEEE, 2014, pp. 398–410.

[23] C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer Science+Business Media. New York, 2006.

[24] Kass, G. V. *An exploratory technique for investigating large quantities of categorical data.* Applied Statistics 29 (2), 119–127, 1980.

[25] Andriy Burkov. *The Hundred Pages Machine Learning Book.* 2019. http://themlbook.com/

[26] Phillips D. *Python 3 Object Oriented Programming.* Packt Publishing. Olton, 2010.

[27] Rackovic Stevo. *Parallel Implementation of Machine Learning algorithms using PyCompsS.* Novi Sad 2018. https://www.dmi.uns.ac.rs/site/dmi/download/master/primenjena-matematika/StevoRackovic.pdf

[28] Breiman, L., Friedman, J.H., Olshen, R., and Stone, C.J., 1984. *Classification and Regression Tree* Wadsworth. Brooks/Cole Advanced Books. Software, Pacific California.

[29] Arik Friedman and Assaf Schuster. *Data mining with differential privacy.* In International Conference on Knowledge Discovery and Data Mining, pages 493–502, 2010.

[30] Xueyang Hu et al. *Differential Privacy in Telco Big Data Platform.* Proceedings of the VLDB Endowment, Vol. 8, No. 12 Copyright 2015 VLDB Endowment 2150-8097/15/08.

[31] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. *A practical differentially private random decision tree classifier*. Trans. Data Privacy, 5 (1):273–295, Apr. 2012.

[32] N. Mohammed, R. Chen, B. C. Fung, and P. S. Yu. *Differentially private data release for data mining*. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11, pages 493–501, New York, NY, USA, 2011. ACM.

[33] Sam Fletcher and Md Zahidul Islam. *Decision Tree Classification with Differential Privacy: A Survey*. CoRR. 2016. http://arxiv.org/abs/1611.01919.

[34] Sam Fletcher and Md Zahidul Islam. 2015a. *A Differentially Private Decision Forest*. In 13th Australasian Data Mining Conference. Conferences in Research and Practice in Information Technology, Sydney, Australia, 1–10.

[35] Zhanglong Ji, Zachary C. Lipton, and Charles Elkan. 2014. *Differential Privacy and Machine Learning: a Survey and Review*. Computing Research Repository (CoRR) 1412.7584 (2014), 1–30.

[36] *PyCOMPSs: Parallel computational workflows in Python*, Enric Tejedor, Yolanda Becerra, Guillem Alomar, Anna Queralt, Rosa M. Badia, Jordi Torres, Toni Cortes, Jesus Labarta. IJHPCA 31 (1): 66-82 (2017). DOI: 10.1177/1094342015594678.

[37] https://scikit-learn.org/stable/

[38] https://www.numpy.org/

[39] https://pandas.pydata.org/

[40] https://seaborn.pydata.org/

[41] https://matplotlib.org/

[42] https://www.scipy.org/

[43] https://www.python.org/

[44] https://www.spyder-ide.org/

[45] https://archive.ics.uci.edu/ml/index.php

[46] http://simpleicon.com/user.html

[47] http://clipartmag.com/detective-girl

[48] https://www.freepik.com/free-icon/question-mark-731610.htm

[49] https://imgbin.com/png/UWRTTEQn/double-question-mark-png

[50] https://nearmyproduct.blogspot.com/2018/07/world-5th-richest-person.html

[51] https://privacy.google.com/?hl=en

[52] https://en.wikipedia.org/wiki/File:User-icon-2.svg

UNIVERZITET U NOVOM SADU
PRIRODNO-MATEMATIČKI FAKULTET
KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj:
RBR
Identifikacioni broj:
IBR
Tip dokumentacije: monografska dokumentacija
TD
Tip zapisa: tekstualni štampani materijal
TZ
Vrsta rada: master rad
VR
Autor: Jelena Novaković
AU
Mentor: dr Dragana Bajović
MN
Naslov rada: Ocuvanje privatnosti u obradi podataka: Istrazivanje o primeni diferencijalne privatnosti u stablima odlucivanja
NR
Jezik publikacije: engleski
JP
Jezik izvoda: e
JI
Zemlja publikovanja: Republika Srbija
ZP
Uže geografsko podrucje: Bačka
UGP
Godina: 2019.
GO
Izdavač: autorski reprint
IZ
Mesto i adresa: Novi Sad, Trg Dositeja Obradovića 4
MA
Fizički opis rada: 8 poglavlja, 78 strana, 52 lit. citata, 18 grafika, 3 listinga, 2 algoritma
FO
Naučna oblast: matematika
NO

Naučna disciplina: primenjena matematika
ND
Ključne reči: Mašinsko učenje, Ocuvanje privatnosti, Algoritmi, Implementacija, Diferencijalna Privatnost, Klasifikacija, Stablo odlučivanja
UDK
Čuva se: u biblioteci Departmana za matematiku i informatiku, Prirodno-matematičkog fakulteta, u Novom Sadu
CU
Važna napomena:
VN
Izvod:U ovom radu prikazujemo implementaciju dva algoritma diferencijalne privatnosti u okviru stabla odlucivanja. Algoritame testiramo na nekoliko stvarnih javno dostupnih skupova podataka i poredimo rezultate.
IZ
Datum prihvatanja teme od strane NN veca: 05.04.2019.
DP
Datum odbrane:
DO
Članovi komisije:
KO
Predsednik: dr Srdjan Škrbić, redovni profesor
Mentor: dr Dragana Bajović, docent
Član: dr Dušan Jakovetić, docent

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
KEY WORDS DOCUMENTATION

Accession number:
ANO
Identification number:
INO
Document type: monograph type
DT
Type of record: printed text
TR
Contents code: master thesis
CC
Author: Jelena Novaković
AU
Mentor: PhD Dragana Bajović
MN
Title: Data mining with privacy guarantees: Decision trees with differential privacy
XI
Language of text: English
LT
Language of abstract: e
LA
Country of publication: Republic of Serbia
CP
Locality of publication: Backa
LP
Publication year: 2019.
PY
Publisher: author's reprint
PU
Publ. place: Novi Sad, Trg Dositeja Obradovica 4
PP
Physical description: 8 sections, 78 pages, 52 references, 18 figures, 3 listings, 2 algorithms
PD
Scientific field: mathematics
SF

Scientific discipline: applied mathematics
SD
Key words: Machine Learning, Privacy Preservation, Algorithms, Differential Privacy, Classification, Decision Tree
UC
Holding data: Department of Mathematics and Informatics' Library, Faculty of Science, Novi Sad
HD
Note:
N
Abstract: In this work, we present two implementations of differential privacy mechanisms within decision tree algorithm. We describe and compare the results of the algorithms on several real public datasets.
AB
Accepted by the Scientific Board on: 05.04.2019.
ASB
Defended:
DE
Thesis committee:
DB
President: PhD Srdjan Škrbić, full professor
Mentor: PhD Dragana Bajović, assistant professor
Member: PhD Dusan Jakovetić, assistant professor