Master's thesis
Master's Programme in Data Science

Differential Privacy applied to Random Forest Classification

Sini Suihkonen

November 11, 2023

Supervisor(s): Razane Tajeddine

Examiner(s): Antti Honkela

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Sini Suihkonen

Työn nimi — Arbetets titel — Title

Differential Privacy applied to Random Forest Classification

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	November 11, 2023	62

Tiivistelmä — Referat — Abstract

The importance of protecting sensitive data from information breaches has increased in recent years due to companies and other institutions gathering massive datasets about their customers, including personally identifiable information. Differential privacy is one of the state-of-the-art methods for providing provable privacy to these datasets, protecting them from adversarial attacks.

This thesis focuses on studying existing differentially private random forest (DPRF) algorithms, comparing them, and constructing a version of the DPRF algorithm based on these algorithms. Twelve articles from the late 2000s to 2022, each implementing a version of the DPRF algorithm, are included in the review of previous work.

The created algorithm, called $DPRF_{thesis}$, uses a privatized median as a method for splitting internal nodes of the decision trees. The class counts of the leaf-nodes are made with the exponential mechanism. Tests on the $DPRF_{thesis}$ algorithm were run on three binary classification UCI datasets, and the accuracy results were mostly comparable with the two existing DPRF algorithms $DPRF_{thesis}$ was compared to.

ACM Computing Classification System (CCS):
Computing methodologies → Machine learning → Machine learning approaches → Classification and regression trees
Security and privacy → Database and storage security → Data anonymization and sanitization

Avainsanat — Nyckelord — Keywords

Differential privacy, Random Forest, machine learning

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

The creation of computers inspired different institutions from corporations to governments to start collecting data about people in the digital format [47]. The practice has only accelerated in the modern day as digitalization has progressed. This has resulted in large datasets, that these institutions use to improve their services *e.g.* through the means of data analytics and machine learning. The benefits of the practice are irrefutable, but these datasets often contain sensitive information about the participants who provided their information. This makes releasing useful information and tools, like statistics, graphs and machine learning models trained on the sensitive data, susceptible to adversaries who aim to uncover sensitive information from the released data. This can lead to significant monetary loss and reputational consequences for the data collector organizations [3]. This threat has elevated data privacy preservation to an important issue and topic of research in the modern day.

Differential privacy (DP) is a state-of-the-art mathematical mechanism that provides provable and rigorous privacy guarantees to sensitive data [47]. It is often used in machine and deep learning algorithms to protect the sensitive training data from information breaches. It has various existing implementations in machine learning algorithms, and one of the algorithms the mechanism has been applied to is the random forest. This algorithm is often used in different tasks from data mining to working with large datasets [41]. Random forests consist of decision trees, which have also been broadly researched in the context of differential privacy [47]. They were also the earliest algorithms to be experimented with in applied differential privacy.

The motivation and research question of this thesis is to examine different existing implementations of differentially private random forest algorithms (abbreviated as DPRF), determine how they avoid the problems and drawbacks inherent to differential privacy, and implement a version of a differentially private random forest based on the background research of existing work. Finally, a comparison of chosen existing DPRF algorithms is conducted against the constructed implementation with experiments on three UCI datasets. Since the source code of these algorithms is not always available, the comparisons will largely focus on the accuracy results presented in the articles.

The implemented random forest will be designed for classification instead of other

possible uses of the random forest, *e.g.* regression. Chapter 2 will go through the relevant background in the context of differential privacy, chapter 3 will focus on how decision trees and random forests function, chapter 4 consists of an extensive review of existing implementations of the differentially private random forest algorithm, and chapter 5 will introduce an implementation of an algorithm based on chapter 4. Finally, the conclusion of the results will be discussed in chapter 6.

2. Differential Privacy

This section will provide an overview of differential privacy and its concepts in the extent relevant for this thesis.

2.1 What is Differential Privacy

Differential privacy (DP) is a method of protecting sensitive information in a dataset from adversaries [47]. Sensitive information in a dataset can be anything from medical records to bank account details, with the common characteristic of creating a privacy risk to the dataset participant if the information is exposed in a successful attack. In essence differential privacy is a robust definition of privacy that ensures that the ability of an adversary to inflict harm should not depend on any individuals presence or absence in the dataset [10]. It provides a theoretical proof of a decreased probability of data breach: even if an adversary knows all other records (rows) of a sensitive dataset, except for one, they still can not deduce any information about the remaining record [47]. Dwork et al. [9] presented preliminary work on differential privacy in 2006. The first steps towards the method were presented already in 2003 [47].

Simple anonymization techniques, such as simply removing personal identifiers from the data, are not sufficient to protect a published dataset from adversaries [47]. An adversary can, for example, link the dataset with available background information, aka. other external datasets. This type of an attack is called a linkage attack, and the external background information is called an auxiliary dataset. One famous example of a linkage attack involves Netflix: in 2006, the company offered a million dollar prize for improving their recommendation system based on a released training dataset of movie ratings of anonymized users [34]. Some time later, the University of Texas researchers Narayanan and Shmatikov showed in [34] that the anonymized Netflix Movie rating data could still be cross-referenced with user ratings publicly available in the International Movie DataBase (IMDB). The user information in IMDB was less anonymized, which led to pieces of sensitive information, such as political stance and sexual identity, to be revealed from the identified users.

Privacy attacks can also be targeted at machine learning models trained on sensitive

data. One of these types of attacks is called a model inversion attack, introduced by Fredrikson et al [17]. In model inversion attack, the adversary aims to infer sensitive information about the dataset participants by making use of the outputs of the machine learning model. As an example, in [17] the inversion attack used black-box access to a public linear prediction model to estimate genotype aspects of the individuals in the data. The outputs of the public model were used as training data for another model, that had the goal of predicting the attributes in the input data. Besides model inversion attacks, there exist many other types of privacy attacks against machine learning models trained on sensitive data, such as membership inference attacks, memorization attacks, hyperparameter stealing, and property inference [23].

It is of best interest of data holders to ensure the privacy of the dataset participants to *e.g.* avoid reputational damages [3]. Differential privacy is one solution to the problem of making sensitive datasets and machine learning models trained on these datasets provably private [47]. Unlike privacy preserving techniques preceding it, like k-anonymity, DP can resist linkage attacks, as well as privacy attacks against machine learning models, like model inversion attacks.

2.2 Theory of Differential Privacy

The formal definition of differential privacy begins with defining two neighbouring databases, D and D' (neighbouring meaning that they differ only in the presence of one row), and a randomized mechanism K that adds noise to the data [47].

Definition 1 *(Differential privacy [47]) The mechanism K provides (ϵ, δ)-differential privacy for all possible output subsets $S \subseteq Range(K)$ and neighbouring datasets D and D', if*

$$P[K(D) \in S] \leq e^{\epsilon} * P[K(D') \in S] + \delta. \tag{2.1}$$

$P[K(D) \in S]$ is the probability of obtaining the result of some query over the dataset D, for example, the mean age of people with some medical condition. The core idea is that the ratio of probabilities $P[K(D) \in S]/P[K(D') \in S]$ should be so similar that an adversary can not tell the difference between them in a way that would compromise the privacy of the dataset participants.

Figure 2.1 represents this idea visually. Dataset D has X_n records and dataset D' has X_{n-1} records [47]. The adversary is assumed to know all the records in D'. When an adversary makes query f, inquiring for example about the mean age of dataset participants with a medical condition, there is a very high probability of the query outputting the same result s on both datasets D and D'. Therefore the query results over D and D' are nigh

indistinguishable from each other, and no additional information of record X_n is leaked. If this logic applies to all the records in dataset D, all of its records are protected.

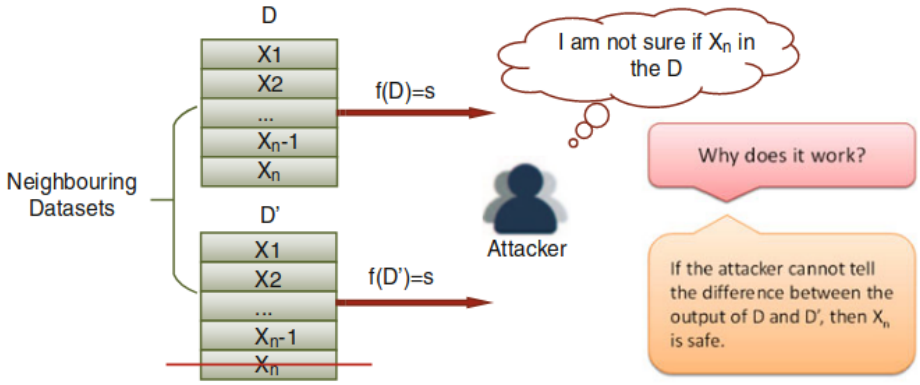


Figure 2.1: Visualization of Differential privacy. Image from Zhu et al. [47]

The ratio of the two probabilities in definition (1) is bounded by e^{ϵ}, where the ϵ is called a privacy budget [47]. It determines the level of privacy provided by the mechanism K. The smaller the ϵ, the stronger the privacy guarantee. When $\delta = 0$, the definition (1) is for *pure* ϵ-differential privacy, which provides stricter privacy guarantees than (ϵ, δ)-differential privacy. The δ in definition (1) represents the probability of an accidental information leak [25]. Its value is required to be small, preferably under $1/|D|$, for the leak probability to be sufficiently small [42].

Due to the randomness of the mechanism K in definition (1), an answer to a differentially private query over the data is a distribution instead of the direct answer to the query [47]. Figure 2.2 demonstrates this visually. For a query output, the ratio of the two probabilities is bounded by e^{ϵ}.

When queries are processed on the data, the privacy budgets ϵ of these queries are summed together under certain conditions [16]. Related to this are two important definitions for differential privacy: composition theorem and parallel composition theorem.

Theorem 1 (The composition theorem) *The composition theorem states that the application of ϵ_i-differentially private queries Q_i satisfies $\sum_i \epsilon_i$-differential privacy [14].*

Theorem 2 (The parallel composition theorem) *The parallel composition theorem states that if dataset D is divided into M disjoint subsets D_i, $i \in [M]$ and queries $Q_i(D_i)$ satisfy ϵ-differential privacy, $\sum_i Q_i(D_i)$ also satisfies ϵ-differential privacy [14].*

The parallel composition theorem therefore states that if the same query function is processed with multiple disjoint subsets of the data without records that overlap, the privacy budget does not need to be summed like in the definition of the composition theorem [16].

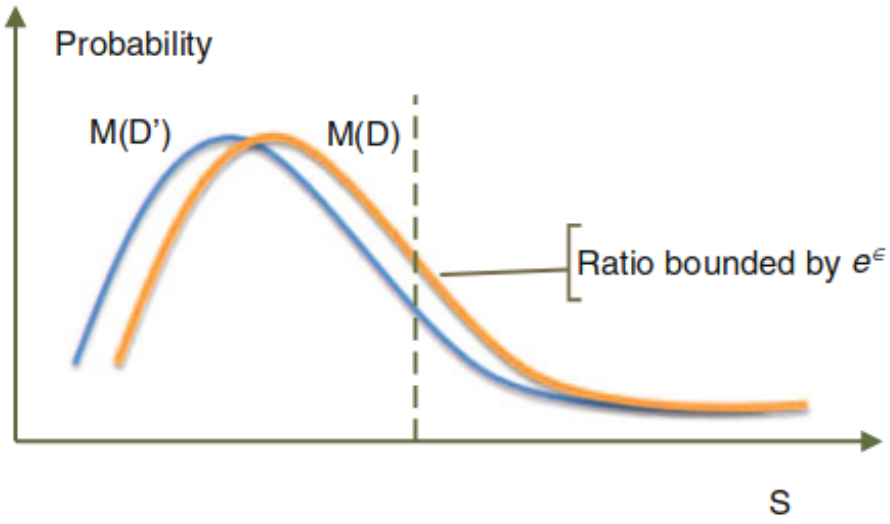


Figure 2.2: Query answer distributions under Differential privacy. The randomized mechanism K is here marked with M instead. Image from Zhu et al. [47]

To summarize, differential privacy is a definition that mechanisms dataset queries are designed to in order to provide privacy to the query result [16]. The randomized mechanism K in definition (1) makes it possible to query a dataset applying differential privacy [16]. The mechanism K takes the database query function as input, making the outputs comply to differential privacy.

In the above examples, the differential privacy technique has only been applied directly to queries over datasets. DP can be applied in machine learning algorithms by implementing a mechanism K that complies to definition (1) to some part of the algorithm, where it is critical to protect the data [47]. These mechanisms, like the Laplace and Exponential mechanism, are discussed in the next section.

2.3 Differentially private mechanisms

A rudimentary example of a differentially private mechanism is a coin toss based mechanism called randomized response [10]. Randomized response is easiest to explain through a practical example: a questionnaire aims to find the approximate percentage of people in a group that have committed a certain type of crime. The question can be put as follows: ”Have you engaged in this illegal behavior this month?”. Using the randomized response, the respondent is instructed to answer according to these steps:

1. Toss a coin
2. If Tails, give the true answer
3. If Heads, toss the coin again. Respond ”Yes” if Heads, and ”No” if Tails

This mechanism is meant to produce plausible deniability for the questionnaire

respondents. The above mechanism is also provably differentially private with $\epsilon = \ln 3$. A case analysis of a chosen respondent shows that the conditional probabilities of the answers are

$$\begin{aligned} Pr(Response = Yes|Truth = Yes) &= 3/4 \\ Pr(Response = Yes|Truth = No) &= 1/4 \\ Pr(Response = No|Truth = No) &= 3/4 \\ Pr(Response = No|Truth = Yes) &= 1/4. \end{aligned} \tag{2.2}$$

With these results we obtain the ratio:

$$\frac{Pr(Yes|Yes)}{Pr(Yes|No)} = \frac{Pr(No|No)}{Pr(No|Yes)} = \frac{3/4}{1/4} = 3 = e^{ln3}. \tag{2.3}$$

The above privacy mechanism works for cases like the described questionnaire use-case, but it does not map to numerical queries. Laplace mechanism is commonly used to make numerical queries differentially private. To explain how this mechanism works, it is first necessary to understand the concept of sensitivity.

2.3.1 Sensitivity

Sensitivity defines the amount of perturbation required in the mechanism K for the privacy bound to apply in definition (1) [47]. If a response for a query f is returned for a dataset D, it determines the noise volume for $f(D)$.

Global sensitivity only depends on the query type, as it considers the maximum difference between the query results on two neighbouring datasets [47]. More formally, for a query $f : D^n \rightarrow \mathbb{R}^d$, the global sensitivity of a query f is defined as [36]:

$$GS_f = \max_{X,Y:d(X,Y)=1} ||f(X) - f(Y)||_1. \tag{2.4}$$

Here, X and Y refer to two different datasets, and $d(X, Y)$ refers to a Hamming distance between X and Y. The Hamming distance is the amount of rows in which the two datasets differ from each other: $d(X, Y) = |\{i : X_i \neq Y_i\}|$. If $d(X, Y) = 1$, they only differ in one row, making X and Y neighbours.

The global sensitivity is good to use when the queries have lower sensitivity values, such as sum and count queries [47]. For other queries, such as medians, it is better to use local sensitivity.

Local sensitivity, compared to the global version, takes both the database instance X and the query f into consideration in its calculations. For a query f: $D^n \rightarrow \mathbb{R}^d$, $X \in D^n$, the definition of local sensitivity [36] for f at instance X is

$$LS_f(X) = \max_{Y:d(X,Y)=1} ||f(X) - f(Y)||_1. \tag{2.5}$$

The key difference between global and local sensitivity is that global sensitivity inspects all the possible pairs of the two neighbouring datasets in the domain of the query f, whereas local sensitivity only inspects a subset of the possible pairs. It should be noted that as local sensitivity is more changing compared to the global version, using it directly may result in information disclosure [47]. Therefore its value is instead changed smoothly with a smooth upper bound. Global sensitivity is more prevalent in literature, and therefore without any specifications, sensitivity often refers to the global version.

l_1- and l_2-sensitivity determine how accurately numerical queries $f : D^n \rightarrow \mathbb{R}^k$, where the queries map databases to k real numbers, can be answered [10].

The l_1-sensitivity of a function $f : D^n \rightarrow \mathbb{R}^k$ is defined as

$$\Delta f = \max_{X,Y \in D^n, ||X-Y||_1=1} ||f(X) - f(Y)||_1. \tag{2.6}$$

In essence, l_1-sensitivity determines the magnitude by which a single individual's data is allowed to change the query function f. This can be understood as the uncertainty that needs to be added to the query output to hide the participation of a single dataset participant.

The l_2-sensitivity of a function $f : D^n \rightarrow \mathbb{R}^k$ is defined in turn as

$$\Delta_2(f) = \max_{X,Y \in D^n, ||X-Y||_1=1} ||f(X) - f(Y)||_2. \tag{2.7}$$

The notation of $||.||$ refers to the mathematical definition of a norm [8]. A norm is applied to a vector of values, and in machine learning they are used to express distances. Given that x_i is the i^{th} element of the vector x, the l_1 -norm (also called the l_1 -distance) is defined as $||x||_1 = \Sigma_i |x_i|$. The simplest and one of the most well known norms is the Euclidean distance, denoted by $||x||_2 = \sqrt{\Sigma_{i=1} x_i^2}$. This norm exemplifies the distance of two points in space, and is also called the l_2 -norm, or the l_2 -distance.

Other types of sensitivity exist as well, depending on the type of the norm that is used: for $p \geq 1$, l_p -sensitivity of query is defined as the maximal l_p -distance between the outputs of the two neighbouring datasets differing in one record [46]. The generalized formula of a norm, called the p-norm, is defined as $||x||_p = (\Sigma_{i=1}^n |x_i|^p)^{1/p}$ for a vector x = $(x_1, ..., x_n) \in \mathbb{R}^n$ and $p \geq 1$ [18].

It is therefore evident that l_1-sensitivity is equal to the sum of the element-wise sensitivities. For example, a vector-valued function f that returns a length-k vector of results that have the sensitivity of 1 has the l_1-sensitivity of k [35]. Using the same logic, a vector-valued function f returning a length-k vector of 1-sensitive results has l_2-sensitivity of $\sqrt{k}$.

2.3.2 Laplace, Exponential and Gaussian Mechanism

To apply differential privacy to numerical queries, one of the mechanisms that can be used is the Laplace mechanism. The Laplace mechanism adds noise sampled from the Laplace distribution to the result of a numerical query function f [47]. The scale of the noise is calculated with the l_1-sensitivity (equation 2.6) of the function f, divided by the privacy parameter ϵ. The mechanism is defined by

$$K(D) = f(D) + Laplace(\Delta f/\epsilon) \tag{2.8}$$

where K is the randomized mechanism, $f(.)$ is the function and ϵ is the privacy parameter. The Laplace mechanism provides strict ϵ-differential privacy for real numeric queries.

Another mechanism K that can be used to make a query differentially private is the exponential mechanism [16]. The definition of this mechanism begins with denoting Δ as the sensitivity, and defining a scoring function $u(z, X) \to \mathbb{R}$. The u receives higher values for the preferable outputs $z \in Z$. The query Q satisfies ϵ-differential privacy if the probability of output z is proportional to $\exp(\frac{\epsilon u(z,X)}{2\Delta(u)})$:

$$Pr(Q(x) = z) \propto \exp(\frac{\epsilon * u(z, X)}{2\Delta(u)}). \tag{2.9}$$

The exponential mechanism is suitable for non-numeric queries [47]. It returns an output without added noise (like a discrete attribute value) by randomizing the results with the mechanism, probabilistically selecting an element from an attribute set based on the scores calculated by the scoring function.

A third known mechanism for differential privacy is the Gaussian mechanism, which like the Laplace mechanism is suited for numerical queries [47]. It can be used to achieve (ϵ, δ)-DP, meaning that the relaxation term δ is above zero in the definition of DP (definition 1). Also instead of l_1 sensitivity like in the Laplace mechanism, l_2-sensitivity (equation 2.7) is used instead. With function $f : D^n \to \mathbb{R}$ over a dataset D, $\epsilon \in (0, 1)$ and $c^2 > 2ln(1.25/\delta)$, when the Gaussian mechanism samples noise to the outputs from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$ with $\sigma \geq c\Delta_2 f/\epsilon$, it provides (ϵ, δ)-DP [10].

2.4 Differential privacy in Machine Learning

There are multiple different approaches to design differentially private machine learning models [24]. For example, the model can be learned on noiseless data, with the DP-mechanism implemented at some chosen stage of the algorithm to add noise to the model. The approach where the training of the model is done without perturbation, but the randomized mechanism K adds noise to the model outputs, is called output perturbation.

In objective perturbation, the mechanism adds noise to the target function (the function mapping input data to a desired output), and then uses *e.g.* the maximum of the perturbed function as the model for outputs. Multiple other perturbation types exist as well, such as input, gradient and prediction perturbation [22].

There are also some key points that should be taken into account when a differentially private machine learning algorithm is being designed. Firstly, the size of the total privacy budget B is relevant, since it determines the algorithm's overall privacy constraints [16]. Second, the necessary amount of dataset queries should be decided, since as the queries accumulate, the total privacy budget B needs to be divided amongst them. The smaller the divided budget ϵ gets, the noisier the query outputs will be. Thirdly, the query sensitivity can be an obstacle: if the query is highly sensitive to individual records, a lot of noise has to be added to the outputs, making the query impractical in the differentially private setting. Finally, the size of the dataset is relevant because the relative amount of noise added to the data decreases as the dataset size increases.

Differential privacy applied to machine learning has also been examined from a critical point of view [4]. Many existing DP-ML applications aim to create privacy preserving models with high accuracy, working around the constraints inherent to the DP method. Differential privacy only works as intended when it is implemented properly: with unsafe privacy parameters, the DP implementation does not provide sufficient privacy. This for example refers to the use of too high values of the privacy budget, with the aim of increasing the model utility. Blanco-Justicia et al. [4] argue that compared to DP-ML, methods that prevent overfitting in machine learning provide higher accuracy, comparable protection and a lower computational cost. They also argue that theory sufficient differential privacy, that uses $\epsilon \leq 1$, is unsuitable for machine learning due to the low accuracy results.

2.5 Other methods of data anonymization

When the machine learning model is designed to be privacy-preserving, the research and techniques generally focus either on the first two phases, preprocessing and training steps, or the last two phases, deployment and serving, of the machine learning pipeline [42]. The goal of this thesis is to look into the algorithm design side of the machine learning, so the first two phases and the anonymization techniques relevant to them (other than differential privacy) will be the focus in this section.

From the approach of anonymization related to the first stage, data preprocessing, one approach is to focus on traditional privacy protection mechanisms, like k-anonymity, l-diversity and t-closeness [42]. These techniques are elimination based approaches to data-anonymity. The goal of k-anonymization is to ensure that all private information

provided about an individual in the dataset can not be distinguished from at minimum of $k-1$ other similar individuals in the data. This is done by removing identifiers from the data attributes and obscuring the quasi-identifiers of the data attributes. Identifiers are information about an individual that fully identifies them (full-name, social security number, etc.) and quasi-identifiers do not directly identify an individual (gender, age, religion, etc.), but combined they can become an identifier [40]. These quasi-identifiers are therefore obscured in k-anonymized datasets [42]. For example, the ages in the dataset could be expressed in brackets of 10, so the age of 27-year-old dataset participant would be expressed as 20-30 year-old in the dataset. This reduces attribute groupings smaller than the size of $k-1$ in the final dataset. A group of at least k records in a k-anonymized dataset that all have the same quasi-identifier values is called an equivalence class [27].

The l-diversity mechanism extends on k-anonymity by also ensuring the diversity of the equivalence classes [42]. Equivalence class is l-diverse if it has a minimum of l well-represented values for the sensitive attributes. L-diversity therefore reduces the data granularity and maintains sensitive field variety. For example, if an attacker knows that a target individual has their gender, age and religion in a dataset containing information about a disease, if all the records of this equivalence class have the same disease, the attacker still extracts this information of the individual out of the data despite the k-anonymity. L-diversity ensures that these row combinations are removed, and that the disease attribute column is diverse enough.

T-closeness is an extension of l-diversity that also adds constraints on the equivalence class value distribution. The equivalence class complies to t-closeness if the distance between the sensitive attribute's distribution inside the equivalence class and its distribution in the whole data is under a threshold t.

The main weakness of these traditional privacy protection methods is that all of them are compromised by uncontrolled background information [47]. Despite the provided group privacy measures, an adversary has the possibility of figuring out what anonymization techniques were used on the published dataset, and use this information for de-anonymization purposes. If the adversary knows that k-anonymization was used, the minimal equivalence classes can be used to reverse the anonymization. This is a version of a linkage attack, called a minimality attack. Another attack that exists against the traditional methods is called a composition attack: the adversary collects extensive background information, and finds out if the published target dataset is vulnerable to it through linkages.

3. Random Forest Classification

This chapter will briefly introduce the basics of random forest classification. Some preliminary concepts required to understand the logic behind the method will be explained first, starting from decision trees. For the scope of this thesis, this introduction will focus on classification tasks instead of *e.g.* regression forests.

3.1 Decision Trees

A decision tree is a non-parametric algorithm for supervised machine learning, suitable for both classification and regression tasks [16]. Nonparametric means that the model does not require the use of parameters based on the underlying data distribution. The input to the algorithm is divided recursively into smaller regions [2].

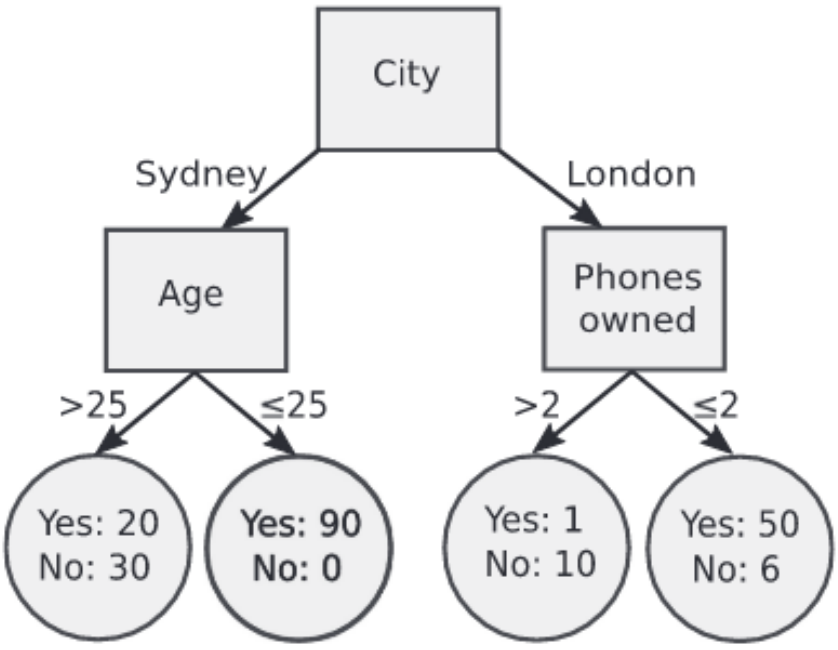


Figure 3.1: A simple example of a decision tree, with d=3 depth. Image taken from [16].

A visual representation of a decision tree can be seen in figure 3.1. Decision trees consists of internal nodes and leaf nodes [2]. The internal nodes use a test function that

splits the input data to branches, and the input data travels down the tree from a root node, being classified to child nodes by node-specific test functions. When reaching a leaf node, this node defines the corresponding class output for the data. A leaf node is a localized region of the input data space, determined by the chain of test function decisions that lead to it. The instances ending up in this region receive the same labels in classification tasks.

As can be seen from figure 3.1, a decision tree is an acyclic directed graph, constructed by dividing the dataset recursively from top to bottom [16]. The test function that splits the data checks the values of the records in a node by a chosen splitting attribute, and then directs the records down to a corresponding child node according to the attribute value. The attribute that creates the highest class homogeneity in the data subsets it produces is selected as the splitting attribute. If the splitting attribute is continuous, the records are split based on whether the values are above or below the best found threshold splitting value. Discrete attributes are commonly split by finding a single value for splitting the attribute, and dividing the parent node data based on whether the record attribute is of that value or not. Another way is to simply create a child node for each unique value of the attribute.

Two relevant methods for evaluating are called *support* and *confidence* [16]. The larger the values these metrics receive, the more trustworthy the node is. Support is the amount of records a node contains, and confidence, or *purity*, is the class label similarity of the node's records, measured as the record percentage of the most common class label. The class in the dataset has multiple (at least two) values, often written as $c \in C$, where c are called labels, or class values.

The decision tree is built by recursively splitting the training data, but eventually the tree needs to stop growing (data stops being split to further leaves). The stopping is achieved by decision tree algorithms by using different kinds of criteria [39]. Some common stopping rules that are used in decision trees are:

1. all records in the node belong to the same class c,
2. a maximum depth parameter is used, and it is reached by the decision tree,
3. the node is not split further because the support of one or more child node would be too small.

Another important step for decision tree algorithms is pruning, referring to removing untrustworthy leaf nodes from the model [16]. When the tree is ready, new records can be given to it, and the test functions will pass the data down to the leaf nodes. The classification of the new input is determined by the most common class label of the training data in the leaf node.

3.1.1 Impurity measures

In classification tasks, the quality of the dataset split in a tree node is quantified by impurity measures [2]. These measures can also be called objective or splitting functions [16]. These measures have to do with the concept of node confidence: the cleaner the data split is, the more accurately all the instances belong to the class determined by it [2]. To demonstrate this mathematically, if a number of instances N_m reach a node m, and N_m^i of N_m instances are of class C_i, the probability of class C_i is

$$P(C_i|x,m) = p_m^i = \frac{N_m^i}{N_m}. \tag{3.1}$$

Here the data instances are denoted by x. The node m is pure if p_m^i are all either class 0 or 1. The result is 0 when all the instances in node m are not of class C_i, and 1 if they all are of class C_i.

One common measure of impurity is called entropy. It can be written as:

$$H(S) = -\sum_{i=1}^{k} p_i log_2(p_i), \tag{3.2}$$

where S corresponds to a set of examples, divided to classes $C_1, C_2, ..., C_k$, and p_i correspond to the probability that an instance set from S belongs to class C_i [11].

Another common impurity measure is the Gini index [31]. This measure calculates the probability of labeling a sample incorrectly when the label is picked at random. Smaller values indicate a smaller probability of mislabeling, and the value will be 0 when all the data points share the same class. The Gini index can be written as:

$$Gini(p) = 1 - \sum_{i=1}^{k} p_i^2, \tag{3.3}$$

where k is the amount of classes and p_i corresponds to the probability that a sample belongs to i^{th} class.

The Gini index can also be expressed in the context of splitting dataset D with an attribute A [28]:

$$Gini\_index(D,A) = \sum_{v=1}^{V} \frac{|D_v|}{|D|} Gini(D_v), \tag{3.4}$$

where v denote the values of A. With this definition, the Gini in (3.3) is rewritten in the context of a dataset D as $Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$, where p_i denotes the i^{th} sample's proportion in D.

Information gain is another impurity measure that can be used in decision tree algorithms to select the most gainful attribute for splitting [11]. The information gain between two random variables X and Y measures the dependency between them: the higher the value, the stronger the dependency. The attribute with the highest information

gain is selected, since it best separates the dataset. The formula for information gain can be written as:

$$InfoGain(A, S) = H(S) - \sum_{j=1}^{n} \frac{|S_j|}{|S|} H(S_j), \tag{3.5}$$

where A is a tested attribute, n is the amount of possible values of A, and S_j are subsets of S that contain objects with the same value of attribute A.

An extension of information gain is called a gain ratio [11]. With A_i denoting an attribute in set A, the formula of gain ratio can be written as [20]:

$$GainRatio(A_i, S) = \frac{InfoGain(A_i, S)}{SplitInformation(A_i, S)}. \tag{3.6}$$

When the splitting attribute A_i creates J partitions S^j of the samples S, $n^j = |S^j|$ denotes the number of samples in a partition j, and $p^j = \frac{n^j}{n}$, the split information is defined as [26]:

$$SplitInformation(A_i, S) = -\sum_{j=1}^{J} p^j log(p^j). \tag{3.7}$$

Gain Ratio reduces the bias Information Gain has to categorical attributes with many different values [26]. The attribute with the largest gain ratio value is selected to split the node during tree construction [11].

3.1.2 Relevant Decision Tree construction algorithms

There are various different versions of the decision tree algorithm. Amongst the most common are ID3, C4.5 and CART. Each of them uses a different criterion for node splitting, and have differences in whether they can process continuous data or not [39].

ID3 is a decision tree algorithm used to recursively generate decision trees based on a provided dataset [11]. Although ID3 produces trees efficiently, its disadvantages are *e.g.* over-fitting when the training data consists of limited number of records, and the inability to efficiently classify continuous data. ID3 uses information gain (3.5) to split the internal nodes.

The C4.5 decision tree algorithm was proposed to improve on the previously mentioned disadvantages of ID3 [11]. The training dataset of C4.5 can contain discrete, binary or continuous data attributes. C4.5 extends ID3 by, *e.g.* statistical pruning for size reduction, ability to work with continuous and missing values, and the use of gain ratio (3.6) as the splitting function. The disadvantages of C4.5 are its tendency to produce unreliable trees when the training set is too small, and bad resistance to data variations during tree construction.

The "classification and regression trees" (CART) decision tree algorithm was proposed by Breiman et al. [1]. CART constructs internal tree nodes with only two out-going

edges (binary trees), and it uses the Gini index as the splitting criterion [31]. CART, as the name suggests, can also produce regression trees [39]. CART can handle continuous and categorical attributes, and identify significant attributes.

Mainly two different approaches exist to constructing decision trees: greedy and random [16]. *Greedy* decision trees use a decision-making strategy in tree building that does not consider the long-term consequences, but makes the best decision according to the immediate situation at hand. A greedy tree maximizes an objective function in each node to split the data. Greedy decision trees use splitting functions, like the Gini-index, information gain and the gain ratio. ID3, C4.5 and CART are all examples of greedy decision tree algorithms. On the other hand, *random* decision trees do not use the heuristics of a greedy decision tree at all: instead of using splitting functions, the splitting attribute is selected randomly. These trees are only usable in ensembles, as their individual prediction performances are very poor. The computational cost of random decision trees is lower compared to greedy trees, because there is no need to constantly compute a splitting function output in every node.

To conclude, decision trees have several known advantages over other supervised machine learning methods [16]. For example, compared to many other methods, they are very human interpretable, the computational cost is relatively low, they can handle missing values and find non-linear relationships in the attributes. However, they are likely to overfit, and small changes in the training data can cause large changes in the final tree. These flaws can be mitigated in many ways, like by limiting the tree depth, pruning, or building a tree ensemble.

3.2 Random Forests

One of the weaknesses of decision trees is their sensitivity to small changes in the training data [16]. In the tree building process with a slightly different dataset D_2 compared to the original D_1, different attributes can be chosen as producing the best split in the early nodes, and the result can be a completely different tree compared to the one constructed on the D_1 training dataset. This is one of the main reasons to use an ensemble of decision trees for the machine learning task instead of just one.

The random forest algorithm is an ensemble of decision trees, that combines the class votes of many decision trees when predicting a class label for a new record [2]. The majority vote wins, and up to hundreds of decision trees can be combined into a single classifier [41]. The resulting classifier is also called an ensemble classifier [16].

When constructing a random forest, decision trees are built to their maximum depths without any pruning [41]. This way the model will have less bias: the trees can overfit, but they overfit differently, creating diversity in the individual tree predictions in the

forest. The randomness in *e.g.* selecting the data subsets from the full training data creates resistance to noise, outliers and overfitting, and the model also does not require much data preprocessing.

The random forest algorithm builds the decision trees using a method called bagging [41]. The idea of bagging is to randomly sample training data records to a "bag", which is an abbreviation of bootstrap aggregation. The data sampling is made with replacement, and each bag is used as a training dataset for a decision tree in the ensemble. The data partitioning also requires sampling the attribute set. This is done during node construction by randomly selecting an attribute subset instead of considering all attributes for splitting. The resulting decision trees, with randomness from both the data and the attributes, have different performance when the algorithm is run on the testing dataset.

Random forests are often used when the training dataset is very large and has a large number of input variables [41]. The random forest algorithm is competitive with nonlinear classifiers such as support vector machines and artificial neural networks, but its performance is very dataset dependent.

4. Differentially Private Random Forest Classification

This section will first introduce the basics of differential privacy applied to random forest classification, starting with general comments of the ideas behind this algorithm. The sections after will focus on a review of existing differentially private random forest classification implementations based on twelve articles on the subject, going more into detail of how the algorithm is implemented and studied in practice.

4.1 Random Forests and differential privacy

Privacy of an algorithm can be compromised by its parts that require querying the training data. In decision trees, privacy leaks from the information related to the internal nodes, derived from the splitting function, and the leaf-node class count queries [16]. Since the node splitting and class count query are both based on the training data, they can leak sensitive information.

Querying the training data with differential privacy always requires using some portion of the total privacy budget B [16]. The less data queries a decision tree requires, the less the total privacy budget has to be divided. This leads to larger available privacy budget portion for the queries, that will be able to output more accurate results with less obscuring noise. This is why designing when and where data queries are truly necessary, and where in the tree they will be implemented, is crucial.

The impurity measures, discussed in chapter 3.1, determine how a decision tree node will be split to child nodes. These measures are used by greedy decision trees, and they always require querying the dataset at least once. These queries are called "non-leaf queries", since they are used in all internal nodes of a greedy decision tree [16]. Data querying in the leaf nodes usually involves querying the class counts of the data in the node. This query enables the classification of new data. These queries are in turn called "leaf-queries". For greedy trees, both "non-leaf" and "leaf" queries are compulsory, but for random trees, only "leaf" queries are necessary.

4.1.1 Non-leaf queries - attribute and splitting point selection

To make the non-leaf queries of a greedy decision tree achieve differential privacy, the splitting function can be rephrased as a data query that complies with differential privacy [16]. For example, the Information Gain (3.5), can be made differentially private this way, by rewriting it as

$$InfoGain(x_i, A) = -\sum_{v \in A}(\frac{n_i^v}{n_i}\sum_{c \in C}\frac{n_i^{v,c}}{n_i^v} log_2 \frac{n_i^{v,c}}{n_i^v}), \tag{4.1}$$

where x_i represent the data subset in node i, $v \in A$ the attribute values, $c \in C$ class labels, n_i^v the record count in the node with value v and $n_i^{v,c}$ is the same as n_i^v with the additional condition of having class label c. The two counting queries (Q1 = $n_i^{v,c}$ and Q2 = n_i^v) can be privatized by adding Laplace noise (2.8) to them:

$$\begin{aligned} Q1 : n_i^{v,c} + Lap(1/\epsilon); \forall c \in C, \forall v \in A \\ Q2 : n_i^v + Lap(1/\epsilon); \forall v \in A. \end{aligned} \tag{4.2}$$

This method for privatization was used in an early paper addressing a differentially private ID3 decision tree, where the information gain was rephrased to only require the data queries Q1 and Q2 for picking the best attribute for splitting [5]. The caveat to this approach is that the portion of the total privacy budget allowed to Q1 and Q2 is small, due to the budget costs of Q1 and Q2 being added together, since they use the same data (theorem 1) [16]. With tree depth d and m as the amount of attributes, the privacy budget per query with these restraints is set to $\epsilon_Q = B/2dm$.

Another means to make the node splitting differentially private is to find the best splitting attribute using the Exponential mechanism (2.9) [16]. One way to do this is to query the dataset of the node two times, first by calculating the node's support, and then finding the best splitting attribute with the exponential mechanism. The queried node dataset subsets on the same tree layer are disjoint, so parallel composition (theorem 2) is applied. By the composition theorem (theorem 1), the privacy budgets of the two queries are added together due to the shared node dataset. The per query budget is then set to $\epsilon_Q = B/2d$.

The exponential mechanism determines the probability of an output by using a scoring function u, which is set to be a splitting function. As seen by Equation (2.9), in order to use the exponential mechanism, the sensitivity Δ of the scoring function needs to be determined. For some of the existing splitting functions, determining the sensitivity is not possible. This is the case with the Gain Ratio (3.6). Since the Gain Ratio sensitivity can not be bounded, this impurity measure can not be properly used with the Laplace nor the Exponential mechanism [16].

4.1.2 Leaf queries - predictions

Commonly a counting query is used to determine the classification result of new records that end up to a leaf-node. One parallel query is made to all the leaf nodes in the decision tree, with the output being the leaf-node class counts:

$$\{n_j^c + Lap(1/\epsilon); \forall c \in C\}; \forall j \tag{4.3}$$

where j is the number of leaf nodes, and c represents the class labels. The ϵ used in equation (4.3) depends on the remaining privacy budget after the non-leaf queries.

Since random decision trees do not query the non-leaf nodes at all, in differentially private tree ensembles the budget is only spent on the class count queries. Since greedy and random tree ensembles have been shown to have similar prediction accuracies, the question of whether there is any benefit in using a greedy splits compared to randomness should be considered [16]. In general, random decision trees are a good choice for minimizing the disadvantages of differential privacy [15]. They are also more computationally efficient because they do not require the use of the training data in tree building process [14].

4.2 Previous work

A total of 12 different research papers presenting a version of the DPRF algorithm, all published between 2009 and 2022, will be first briefly introduced below, and then their most important contributions within the scope of this thesis will be discussed in more detail. The contents are discussed in different subsections, from feature selection to dataset queries. Thoughts and observations on the articles will be summarized in the last subsection.

In 2009, Jagannathan et al. [21] were among the first to implement a differentially private random decision tree ensemble classifier. Their version of the algorithm gives acceptable prediction accuracy even with small datasets and without compromising privacy.

Moving to 2014, Patil and Singh [37] study differential privacy in the context of random forests, and the effect of different quality functions on accuracy and noise sensitivity. They use the ID3 decision tree algorithm (described in subsection 3.1.2), recognizing its limits in accuracy due to the use of information gain splitting function.

Three of the papers were published in 2015 [6, 14, 38]. Rana et al. [38] introduce a random forest constructed under the differentially private framework that does not follow strict differential privacy, and instead it keeps the complete data distribution invariant to change in data instances, only keeping the necessary statistics invariant. This approach resulted in higher utility. Fletcher and Islam (2015) [14] propose a DPRF algorithm with differentially private data queries that obtains high accuracy even with high privacy

requirements. This is done with the help of signal-to-noise (SNR) ratio, and using signal averaging to reduce the noise in the queries. Bojarski et al. [6] construct a random forest consisting of O(log(n)) random decision trees, n being the size of the dataset. The trees are random, and the attribute selected for the split is chosen uniformly at random at each given node. Private and non-private decision trees are evaluated with three methods of classification: majority voting, threshold averaging and probabilistic averaging.

Two of the papers are from 2017 [15, 28]. Li et al. [28] propose an algorithm called DPRF-gini, that aims to improve the overall data privacy level of the random forest algorithm. This algorithm uses the exponential mechanism in non-leaf queries, with the Gini index as its utility function, and the Laplace mechanism in leaf class count queries. Fletcher and Islam (2017) [15] seek to improve the quality of the DPRF by reducing the amount of obligatory queries and minimizing their sensitivity. This is done by querying the random decision tree leaf nodes with the exponential mechanism, instead of the state-of-the-art Laplace mechanism.

In their 2019 paper, Hou et al. [20] propose a DPRF that protects the private information in the data classification process. To reduce the accuracy loss inherent to differential privacy, they introduce a hybrid decision tree, which combines the information gain of the ID3 decision tree and the information gain ratio of the C4.5 decision tree, creating a new splitting function that improves the classification accuracy of a single decision tree. They also propose a new privacy budget allocation strategy.

In 2020, Guan et al. [19] presented their implementation of a differentially private greedy decision forest (DPGDF) algorithm. It differs slightly from the standard random forests since it uses sampling without replacement instead of bagging. The main focus of this article is on improving privacy budget allocation to the trees.

Two of the papers were published in 2021 [7, 45]. Zhang et al. [45] propose a DPRF for data mining called DPRFMaxTree that achieves high accuracy. They also propose a new splitting function, F_Max, and use the CART decision tree with the exponential mechanism. Their DPRF also improves the utilization of the privacy budget. Consul and Williamson (2021) [7] propose a differentially private median forest (DiPriMe), a high-utility differentially private random forest. The trees in the ensemble are built using the noisy median of the attribute values, calculated through the exponential mechanism, for splitting the nodes. The motivation for using the noisy median as the splitting function is that it encourages balanced leaf nodes, avoids the need to query small subsets of the data, and ensures that the added noise is not too large.

In the final paper of the twelve, published in 2022, Liu et al. [30] design a two-phase DPRF. They focus on improving the feature selection method of the tree in order to reduce the correlation among individual trees in the ensemble.

4.3 Data and data queries

This section will focus on the different data queries conducted by the DPRF algorithms. The sensitivity of these queries, feature selection and the handling of continuous attributes will be addressed as well.

4.3.1 Continuous attributes

One of the main issues in designing DPRF algorithms is whether and how they can handle continuous attributes [16]. A common means to make continuous attributes usable in decision trees is discretizing them. Discretizing refers to the modification of continuous variables to a discrete form by creating a set of intervals that go across the values of the chosen attribute [43]. For example in [14], continuous attributes of the UCI datasets used in the experiments are discretized by splitting their domains to 5 equal sized bins. In [19] a similar method to partition and discretize continuous attributes is used. The values of a continuous attribute with domain d are sorted in non-descending order, the average of every 5 records is calculated, and the original values are replaced with these average values. The domain size of the attribute is reduced to $|d|/5$.

Other ways to discretize continuous data is to use a splitting function or a decision tree algorithm that can process continuous attributes. For example in [45], the continuous dataset attributes are made discrete with the exponential mechanism through the CART decision tree algorithm. In [7], the proposed DiPriMe random forest algorithm is designed to be able to handle both categorical and continuous attributes through the differentially private median estimate splitting function, used as a scoring function for the exponential mechanism.

More complex discretization methods exist as well. For example in [37], the continuous data is preprocessed with a supervised method called entropy-based discretization. It selects the boundaries for the continuous attribute by calculating the class information entropy for the partitions. An interval with the attribute values is recursively split to sub-intervals. The used splitting function is entropy, and a principle called minimum description length (MDL) is used to determine the stopping condition for the recursion.

The authors in [37] define class entropy by the formula $H_C(D) = -\sum_{c \in C} (\tau_c/\tau) log(\tau_c/\tau)$, where D is the dataset, τ is the size of D ($\tau = |D|$), and C is the class attribute. Information gain (3.5) is used to measure the entropy difference by the presence and absence of a split made with attribute A: $InfoGain(A, D) = H_C(D) - H_{C|A}(D)$, where $H_{C|A}(D) = -\sum_{j \in A}(\tau_j^A/\tau) * H_C(D_j^A)$.

All values of a continuous attribute A are considered as candidate splitting points, V, and the range is split in two by the value of V that has the lowest entropy value.

With a sample set S, the MDL principle stops the splitting when $InfoGain(S, V) = H(S) - H(S, V) < \delta$, where

$$\delta = \frac{log_2(n-1) + log_2(3^k - 2) - [mH(S) - m_1H(S_1) - m_2H(S_2)]}{n}, \tag{4.4}$$

where k is the class count, m is the number of classes in each S_i, n is the total number of samples in S, and S_1 (left split) and S_2 (right split) are the parts of sample set S where the interval boundary V has split the data.

Another more detailed method for discretizing is presented in [28]. It considers a continuous attribute A with n values $a_n \in A$ in dataset D, with a set of the attribute values sorted by size. For all neighboring sections $[a_i, a_{i+1})$, the mean value of the section is selected as a candidate splitting point, and the set of all candidate splitting points is defined as

$$V_a = \{\frac{a_i + a_{i+1}}{2} | 1 \leq i \leq n-1\}. \tag{4.5}$$

All these candidate splitting points are computed using the Gini index (3.4), which gives the optimal splitting point as the result when the following formula is used:

$$\min_{u \in V_a} Gini(D, a, u) = \min_{u \in V_a} \sum_{\lambda \in \{-,+\}} \frac{|D_u^\lambda|}{|D|} Gini(D_u^\lambda), \tag{4.6}$$

where D is the dataset, a the attributes and u the splitting points.

In some cases discretization is not used, but some other method of determining the means of working with a continuous attribute is designed. In [38], if the selected splitting attribute is continuous, a splitting point is found with its values $a \in A$ by finding the average value of a for all the records in a node with class label C_1 and C_2 as follows:

$$\delta = \frac{1}{2}(\frac{1}{N_1} \sum_{n \in C_1} a_{jn} + \frac{1}{N_2} \sum_{n \in C_2} a_{jn}) \tag{4.7}$$

where the splitting point is δ, the features are marked with j, N_1 and N_2 are the data counts for C_1 and C_2. The splitting point calculation is made differentially private using the Laplace mechanism for perturbation.

4.3.2 Splitting functions

Another key problem to solve when designing decision trees is optimally selecting the splitting feature and the splitting point for splitting a node to child nodes [30]. The splitting feature is selected from the dataset (for example age or gender) and the splitting point in turn is one of its values, or a point from the range of the feature values, if the attribute is continuous (age: 35, gender: Male).

Many of the articles use one of the standard splitting functions, introduced in section 3.1.1. For example Li et al. [28] and Guan et al. [19] both use the Gini index, and Rana et al. [38] use the information gain. The decision trees in these articles are therefore, by definition, greedy.

Some of the articles instead design an entirely new splitting function to improve the algorithm. In [20] the attribute metric IG_GR is constructed. It linearly combines the information gain (3.5) used in ID3 decision tree to select the split attribute, and the information gain ratio (3.6) used in C4.5 decision tree to select the splitting attribute and the splitting point for continuous attributes:

$$IG\_GR(D, A_i) = w_1 * InfoGain(D, A_i) + w_2 * GainRatio(D, A_i). \tag{4.8}$$

Here, D is the training dataset and A_i is an attribute in A, and w_1 and w_2 are weight coefficients, where $0 \leq w_1,\ w_2 \leq 1$ and $w_1 + w_2 = 1$. The IG_GR formula can also be written as:

$$IG\_GR(D, A_i) = (w_1 + \frac{w_2}{SplitInformation(D, A_i)}) * InfoGain(D, A_i). \tag{4.9}$$

The attribute with the largest IG_GR score is selected. This measure was designed to mitigate the issues of the two impurity measures it combines. For example, the information gain favors attributes with larger value sets, and with small information gain values the information gain ratio result can get unreasonably large. In [20], the IG_GR is used as a scoring function for the exponential mechanism.

In [37], alongside the Gini index and the information gain, a splitting function called the max operator is considered as a quality function for the exponential mechanism. The max operator works by selecting the highest frequency class from the input data, and it is written as

$$Max(D, A) = \sum_{j \in A} (\max_c(|D^A_{j,c}|)). \tag{4.10}$$

Since the function value depends on the class frequency, its sensitivity is 1. The study found that there were no significant differences in accuracy in their implementation of the algorithm between the three tested splitting functions. This is notable considering that the three splitting functions have different sensitivities to noise.

In [45], a new splitting function designed as an improvement on the max operator (4.10) is proposed, called F_Max. In addition to the class frequency counts, F_max also takes the weight of the class into account. The measure is defined with supposing a dataset D with n rows, an attribute set A, a random subset of attributes F from A, and a set of m class labels C. For a chosen splitting attribute from a_i from A, there is i values, v_i. In equation (4.11), n_i represents the count of a_i with value v_i, and n_{ic} the count of

value v_i of attribute a_i with class value c. F_Max calculates the sum of products of each maximum class count value of the A attributes and their respective weights. F_max is then used to pick the best split attribute $\hat{A}$ [45]:

$$F\_Max(D, A) = \sum_{c=1}^{m} \frac{\sum n_{ic}}{n_i} * (\max(n_{ic})) \tag{4.11}$$

$$F\_Max(D, \hat{A}) = \max_{F \in A}(F\_Max(D, A)) \tag{4.12}$$

$$\hat{A} = \arg\max_{A}(F\_Max(D, \hat{A})). \tag{4.13}$$

In [7], the presented algorithm, differentially private median (DiPriMe) forest, modifies a non-private tree construction algorithm to be robust to noise addition. The node splitting is done by a privatized estimate of the median value. This splitting strategy is designed to improve balanced leaf-node data occupancy. The sensitivity of the median splitting function is also low, so adding noise to outputs will not hinder the tree construction. The authors argue that many common node splitting methods produce low-occupancy nodes, which increase the effects of noise in the trees. For example, if the branches of a decision tree are learned for each category of the chosen attribute, more low-occupancy nodes are produced compared to a binary tree. Random decision trees in turn avoid the data queries of the internal nodes, but the large leaf-node budget is hindered by a leaf node occupancy that varies much in size due to the randomness of the splitting.

For each splitting query, from picking the candidate splits (4.14, 4.15) to the splitting attribute (4.16), the budget $\epsilon_s = \epsilon_a = (pB)/(2d_m)$ is allocated from the total budget B, for maximum depth d_m and some chosen probability $p \in (0, 1)$.

The median splitting, for both scoring the possible splitting points of the attributes and picking the attribute to split on, makes use of the exponential mechanism. For the continuous attributes in the data, let dataset $D_i = (X_i, Y_i)$ be split on attribute a with range $R_a = [a_L, a_U] \subset \mathbb{R}$. The scoring function for the potential splits $r \in R_a$ is defined as $q(r) = ||X_{i,a} \cap [a_L, r)| - |X_{i,a} \cap [r, a_U]||$, with sensitivity of $q(r) = 1$. With ϵ_s-DP, the probability of selecting r is

$$Pr(r) \propto \exp(\frac{\epsilon_s}{2} * ||X_{i,a} \cap [a_L, r)| - |X_{i,a} \cap [r, a_U]||). \tag{4.14}$$

For categorical attributes, the scoring function is defined as $q(C) = ||C| - |X_i \backslash C||$, with sensitivity $q(C) = 1$. It is calculated for all splits (C, X_i \C) of the attribute, and by determining the maximizing split. With ϵ_s-DP, a split C is chosen by the probability

$$Pr(C, X_i \backslash C) \propto \exp(\frac{\epsilon_s}{2} * ||C| - |X_i \backslash C||). \tag{4.15}$$

For selecting the splitting attribute, the exponential mechanism with negative mean squared error as a scoring function is used. Its sensitivity is $4P^2/N_i$, where the class target value lies in $[-P, P]$. With ϵ_a-DP, splitting on attribute $\hat{a}$ occurs with probability

$$Pr(\hat{a}) \propto \exp(\frac{\epsilon_a * N_i}{8P^2} * \text{MSE}(\hat{a})). \tag{4.16}$$

Splitting using the median is not a recommended approach in non-private random forests, because it produces higher classification errors compared to more sophisticated splitting functions. The method is viable in differentially private random forests due to the resulting balanced trees with less low-occupancy leaf-nodes. Well-balanced splits are beneficial for performance in differentially private decision trees due to the added noise, that can obscure the classification results in leaf-nodes with small data occupancy.

4.3.3 Query Sensitivity

The sensitivity of a data query has effect on the noise introduced to the algorithm, as the sensitivity of a splitting function gives an upper bound on the amount of noise we must introduce to the outputs to preserve privacy [10]. The noise is also affected by the used differential privacy mechanism.

The two mechanisms that appeared in the articles are the Laplace (2.8) and Exponential mechanism (2.9). The articles [20, 19] will be used as a basis to give an example of how these mechanisms can be used together with splitting functions and class count queries. In both articles the process of constructing a decision tree, the dataset D is queried mainly by the leaf-node counting query, and the attribute query, including both the selection of splitting attribute and the attribute specific splitting point in internal nodes.

If the counting query is denoted by f_1 and the attribute query with f_2, and the scoring function used in the exponential mechanism is denoted with $u(.)$, then:

$$f_1(D_i) = \{|D_{i,c}| \forall c \in C\} \tag{4.17}$$

$$f_2(D_i) = \overline{A} \in A : \overline{A} = \arg\max_{A' \in A}(u(D_i, A')), \tag{4.18}$$

were, $D_i \subseteq D$, $i \in A$, $|D_{i,c}|$ is the number of data samples with attribute i and class $c \in C$, and $\overline{A}$ is the selected split attribute. These equations are made private by applying the Laplace mechanism to the counting query Q_1, and the exponential mechanism to the attribute query Q_2:

$$Q_1(D_i) = f_1(D_i) + (Lap(\Delta f/\epsilon))^d = \{|D_{i,c}| + Lap(\frac{1}{\epsilon}); \forall c \in C\} \tag{4.19}$$

$$Q_2(D_i) = \max_{A' \in A}(u(D_i, A')) \propto (\exp(\frac{\epsilon * u(D_i, A)}{2\Delta(u)})). \tag{4.20}$$

In (4.19), since the sensitivity of a class count query Δf is 1, the distribution scale is set to $\Delta f/\epsilon = 1/\epsilon$ [19]. The exponential mechanism in (4.20) outputs the best attribute with a probability, depending on the outputs of the scoring function u. The sensitivity of the scoring function $\Delta(u)$ depends on what splitting function is used.

There are also alternative versions of sensitivity to take into consideration when designing dataset queries. For example in [15] a DPRF with random trees that used smooth sensitivity in the leaf-node queries was proposed. Instead of using class counts and the Laplace mechanism to determine the classification results, the exponential mechanism was used. The exponential mechanism only outputs the most frequent class label in the node. The authors of [15] argue that this way the privacy budget does not get spent on unnecessary information, like when the noise is added to the class counts. Smooth sensitivity is a differentially private version of local sensitivity (2.5), and it is defined as [36]:

Definition 2 *(Smooth sensitivity [36]) With Hamming distance k between two datasets D_1 and D_2, the local sensitivity (LS), of function f is*

$$S^k(D_1) = \max_{D_2 : d(D_1, D_2) \leq k} LS_f(D_2). \tag{4.21}$$

With this, the smooth sensitivity of f can now be written using $S^k(D_1)$:

$$S^*(f, D_1) = \max_{k=\{0,1,\ldots,n\}} e^{-k\epsilon} S^k(D_1). \tag{4.22}$$

It should be noted that in an Addendum of [15] added in 2021, modifications are made to the algorithm due to received criticism about the lack of proof on smooth sensitivity working with the exponential mechanism as a replacement for global sensitivity, as originally proposed in the 2017 article. To achieve pure differential privacy with smooth sensitivity, Cauchy noise should be used [36].

In the 2021 addendum of [15], a solution is proposed, in which the use of smooth sensitivity is removed altogether from the original algorithm. The mechanism used in the solution is called report one-sided noisy arg max (ROSNAM), a version of the exponential mechanism designed for tasks where the goal is to determine which class A or B is more common than the other [10]. If the class frequency is set to 0 for class A, and $f_c > 0$ for class B, the utility now is tied to the class counts. Larger count results have a higher utility and sensitivity $\Delta u = 1$. The utilities for the classes are now defined as A = 0, and B = f_c. The maximum probability of the mechanism outputting the wrong result (the less frequent class A), is $2\epsilon^{-f_c\epsilon/2}$.

ROSNAM adds noise to the class utilities from a one-sided exponential distribution, with parameter $\epsilon/\Delta u$ when the utility is monotonic, and then outputs the argmax result. Since the addition of a record only ever increases the maximum count utility function, the class counts are monotonic [15]. The new utility function replacing the version used in the original work is $u = f_c$, and the sensitivity of the utility function is defined as $\Delta u = 1$, since the utility is a simple count. The original exponential mechanism $\exp(\epsilon u/2\Delta u)$ is also redefined as $\exp(\epsilon f_c)$. The authors argue that the rest of the original algorithm works with these changes implemented.

4.3.4 Attribute selection

Usually during node splitting in random forests, a random subset of the splitting attributes is chosen, and the best splitting attribute is then chosen from this subset [30]. The attribute selection process in [30] is more complex as it aims to improve the quality of this attribute subset to enhance decision tree performance, and it has three phases. With A_{set} as a set of attributes, first, symmetrical uncertainty (SU) evaluates relevance between attribute a_i and class label c for each attribute in A_{set}. The a_i is deleted from set A_{set}, if the SU value indicates that it is not relevant to c. Second, interaction for all the a_i is calculated with the other attributes, a_j, $a_j \in A_{set} - a_i$. The interaction is calculated with normalized interaction gain (NIG). The weights for a_i are also calculated, using an algorithm called IG-RFE. The weight corresponds to how much influence the attribute has in the context of the classification problem [29]. Finally, the remaining attributes are sorted from highest to lowest weight, and top-p of them are chosen for the attribute subset [30].

Before defining symmetrical uncertainty and interaction gain, the concept of mutual information needs to be defined. Joint mutual information (JMI) is a technique used to calculate correlation between random variables and a target [30]. JMI is denoted as I, and defined as:

$$I(X_1...X_n;Y) = \sum_{x_1 \in X_1} ... \sum_{x_n \in X_n} \sum_{y \in Y} p(x_1...x_n, y) * log \frac{p(x_1...x_n, y)}{p(x_1...x_n)p(y)}, \tag{4.23}$$

where $X_1...X_n$ are the random variables, Y is the target. The $p(x)$, $p(y)$ are the distributions of x, y, and $p(x, y)$ is the joint distribution of x and y [29]. If only one random variable is considered, $I(X;Y)$ is called the mutual information [30].

Symmetric uncertainty (SU) is used to estimate the correlation between two random variables X and Y. It is defined as [30]:

$$SU(X,Y) = \frac{2 * I(X,Y)}{H(X) + H(Y)}, \tag{4.24}$$

where $H(.)$ represents entropy (3.2).

Interaction gain, which measures interaction between random variables (X, Y, Z), is defined as

$$IG(X,Y,Z) = I(X,Y;Z) - I(X;Z) - I(Y;Z). \tag{4.25}$$

Normalized interaction gain (NIG) [29] in turn is defined as:

$$NIG(X,Y,Z) = \frac{1}{2} + \frac{IG(X;Y;Z)}{2 * [H(X) + H(Y)]}. \tag{4.26}$$

The IG-RFE algorithm, which calculates the attribute weights in the second step of the selection process, is further described in Lin et al. [29].

4.4 Privacy

Some of the articles, especially ones using greedy decision trees, presented sophisticated strategies to optimally distribute the privacy budget across the forest and the tree structure. In [38], a relaxation to the definition of differential privacy was explored.

4.4.1 Privacy budget allocation

The ability to distribute the privacy budget sensibly across the decision tree layers has several approaches in the relevant literature. In more traditional approaches, in a differentially private decision forest ensemble of t trees, each tree receives a privacy budget of $\epsilon = B/t$ [14, 20, 45]. The division with number of trees t makes the standard deviation of the added noise t times larger [38]. The smaller the budget gets, the more noise is added, reducing the utility of the resulting DPRF algorithm.

The concept of signal-to-noise ratio (SNR) can be used to aid in constructing a privacy budget allocation strategy that improves the forest's accuracy [20]. SNR is a ratio between a signal measurement and the signal's background noise [14]. SNR can be applied when there is no correlation between the signal and the noise, the signal remains unchanged if re-measured (this condition excludes the noise), and the noise in all signal measurements is random, with zero mean and immutable variance value. Mathematically the signal-to-noise ratio can be expressed as

$$SNR = \frac{\mu}{\sigma} \tag{4.27}$$

where μ is the mean of the signal, and σ is the noise's standard deviation. Both μ and σ are expressed in the same unit.

The conditions of SNR are met *e.g.* by the Laplace mechanism. The noise is not affected by the signal, and if the underlying dataset is queried again, the real number of records matching the query (signal) remains the same. The mean of the Laplace

distribution is zero, and the variance is $2x^2$, x representing the scale. From equation (2.8), it can be seen that $x = \Delta f/\epsilon$.

As a simple example of how SNR is used in practice when determining privacy budget scheme, in [7], the Laplace mechanism is used to privatize the leaf-node classifications. The budget allocated to the leaves is denoted as ϵ_l. The utility of the resulting class estimate at i^{th} leaf depends on the privacy budget, and the number of leaf-node data points, N_i. Small values of ϵ_l and N_i lead to the noise dominating the data signal: either ϵ_l per leaf or N_i must be kept high enough to maintain the utility. Specifically, conditions where $\epsilon_l < 1/N_i$ are undesirable, since this indicates that magnitude of the data signal is smaller than the noise magnitude.

SNR can also be used in more complicated privacy budget allocation schemes. In [20] the balanced version of SNR is used to aid a new proposed privacy budget allocation strategy. The total privacy budget is marked with B, the number of decision trees with t, the depth of the decision tree with d and the maximum depth of the tree with d_m. Each tree uses the standard tree-wise privacy budget $\epsilon = B/t$. Data points in nodes on the same layer do not intersect, so the per-layer budget can be used in its entirety for each node query by using theorem 2. The budget per-layer is split evenly between a node's counting and attribute query, and in the last layer the budget is only allocated to the counting queries.

If the maximum depth d_m is reached, the per-query privacy budget for each layer is set to $\bar{\epsilon}$:

$$\bar{\epsilon} = \frac{\epsilon}{2d_m - 1}. \tag{4.28}$$

The denominator is $2d_m$ - 1 since all the layers contain two kinds of queries, expect for the bottom layer, which only has count queries for the leaf-nodes.

The data size and attribute number at each node decreases as the tree grows in depth. The SNR will therefore be unbalanced for different depth data queries if equal privacy budget is added to them, due to deeper nodes having less data in them. This is why in the budget strategy [20], a larger portion of the budget is allocated for the deeper nodes to better protect the data signal.

Balanced SNR is achieved in practice by assigning budget weights to the tree layers. The privacy budget weight for the root node is set to $w_1 = 2/d_m$, for the subsequent i^{th} layers the weight is assigned by the rule $w_i = 2/(d_m - i + 1)$. The layers together sum up to the total share of the privacy budget:

$$w = \frac{2}{d_m} + \frac{2}{d_m - 1} + .. + \frac{2}{d_m - d + 2} + \frac{1}{d_m - d + 1}. \tag{4.29}$$

The per-tree privacy budget is $\epsilon = B/t$, making the unit share privacy budget by

weight $\epsilon_u = \epsilon/w$. The weight-based allocation scheme is illustrated in figure 4.1.

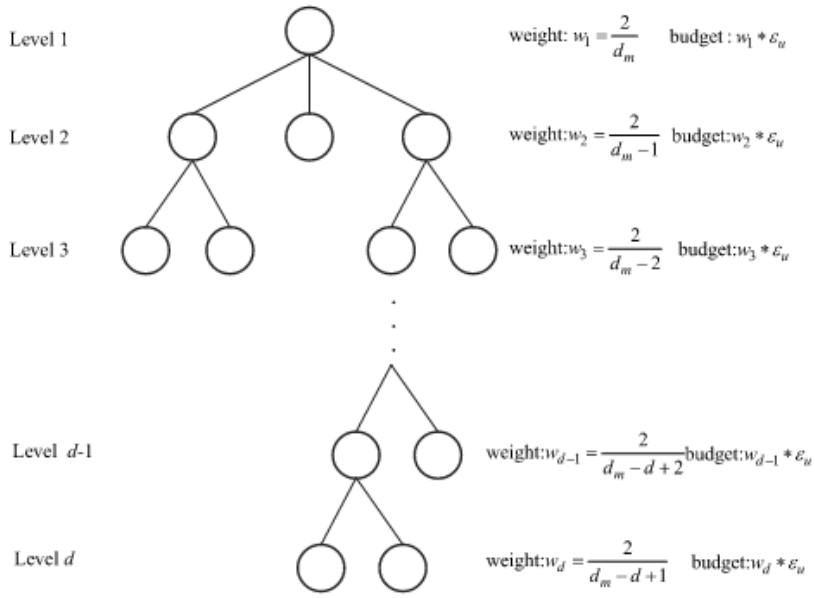


Figure 4.1: Budget allocation strategy by weight. Image from Hou et al. [20].

The privacy budget $\epsilon_i = \epsilon_u * (2/(d_m - i + 1))$ for layer i, $1 \leq i \leq d - 1$, is split between the counting and attribute query to $\epsilon_{i,1}$ and $\epsilon_{i,2}$, while the privacy budget for the deepest layer counting queries is $\epsilon_d = \epsilon_u(1/(d_m - d + 1))$.

The layer's budgets accumulate to the total privacy budget, $w = 2d - 1$. If unit share budget is set to $\epsilon_u = \epsilon/(2d_m - 1)$, the privacy budget used by building the tree is:

$$w * \epsilon_u = \epsilon * \frac{2d - 1}{2d_m - 1} \leq \epsilon. \tag{4.30}$$

In [30], another similar strategy is proposed, which also aims to distribute more privacy budget to the deeper tree nodes. The privacy budget ϵ, denoting the tree-wise budget $\epsilon = B/t$, is used to build a tree which provides ϵ-differential privacy, and d_t represents the current tree depth. The budget ϵ is divided to s_t shares:

$$s_t = \frac{1}{d_m} + \frac{1}{d_m - 1} + ... + \frac{1}{2} + 1. \tag{4.31}$$

The root node receives $\epsilon_1 = \epsilon/s_t * 1/d_m$ of the budget, with the general rule for the layers being allocated

$$\epsilon_{d_t} = \frac{\epsilon}{s_t} \times \frac{1}{d_m - d_t + 1} \tag{4.32}$$

of the budget. No further budget division is done in the algorithm.

In [19], a different privacy budget allocation scheme is developed, with the same goal of allocating more privacy budget to the deeper leaf nodes. In the strategy, the total budget B is split evenly to two parts: ϵ_1 and ϵ_2. The total budget can be used since the

training data for each decision tree is sampled from the full dataset without replacement. The ϵ_1 is divided evenly to the internal nodes, and ϵ_2 is for the leaf-nodes. The next steps of the budget allocation strategy only concern the leaf nodes. The authors state that this is because the exponential mechanism is used in the internal nodes for splitting, and it is not affected by the amount of data in the internal node.

In the strategy, Laplace noise has variance $\text{Var}(\text{Lap}(\epsilon)) = 2 \ / \ \epsilon^2$. The total error in tree T is formulated as $Error(T) = \sum_{i=1}^{d} 2g_i/(\epsilon_{2,i})^2$, where g_i is the i^{th} layer budget, d the tree depth, and $\epsilon_{2,i}$ the i^{th} layer budget for the leaves. The by layer budget allocation problem can be formulated as finding the best result of $\text{Error}(T)$, with the requirement $\sum_{i=1}^{d} \epsilon_{2,i} = \epsilon_2$. In [19], it is proved that $\text{Error}(T)$ is minimized when

$$\frac{\epsilon_{2,i+1}}{\epsilon_{2,i}} = \sqrt[3]{\frac{g_{i+1}}{g_i}}. \tag{4.33}$$

In [19], layer privacy budgets for the leaf nodes are found by setting a condition $g_{i+1}/g_i = 2$. Using equation (4.33) and the equal ratios, the budget $\epsilon_{2,i}$ is formulated as

$$\epsilon_{2,i} = \frac{(1 - \sqrt[3]{2})\sqrt[3]{2^{i+1}}}{1 - \sqrt[3]{2^{d_m}}}\epsilon_2. \tag{4.34}$$

In [38], it is argued that the DPRFs that use the privacy budget allocation scheme $\epsilon = B/t$, *e.g.* [6, 21, 37], high noise is introduced to the trees, leading to poor accuracy. This "worst case scenario" per-tree budget is not necessary, since this strategy assumes the unlikely event that a specific leaf node reoccurs in all trees. The algorithm in [38] uses an alternative approach to differential privacy, that protects the variances of data instance estimations instead of the full distribution of the dataset. This approach is discussed separately in the next section.

4.4.2 Privacy requirement relaxation

In [38], a different approach to protecting the training data is used, by using a relaxation of the definition of differential privacy (definition 1) to only protect the variance of the estimate of data instances. The benefits of this relaxation are an increase in utility, and avoiding unnecessary noise addition that protects against very unlikely attacks. The privacy requirement relaxation prevents attribute and class values from being discovered, but it does not conceal if an individual is in the dataset or not [16]. The scope is also limited to forests using bootstrapped samples, where the trees are constructed for binary classification [38].

The DPRF built on this approach is designed to ensure that with a set upper bound to the forest size, the attribute/class label estimation error variance is not lower compared to its prior variance [38]. The authors demonstrate how the variances are preserved using

a hypothetical adversarial attack. The aim is to demonstrate how many trees should be allowed so that the estimator variances are still protected.

The results for preserving the variance of the attributes is culminated in the following theorem, that gives a bound for when the attribute privacy is still preserved despite auxiliary information.

Theorem 3 (Privacy Preservation of Attributes [38]) *The attribute estimate $\tilde{x}_j$ preserves the attribute variance privacy if the number of trees, t, is bounded by*

$$\begin{aligned} &\frac{t}{\sqrt{M}} + 3\sqrt{\frac{t}{\sqrt{M}}}(1 - \frac{1}{\sqrt{M}}) \leq \\ S_1(1 - \frac{1}{N_1 + 1}) + &\frac{m_1^2}{m_2^2} S_2(1 - \frac{1}{N_2}) + \frac{36}{\epsilon^2} \end{aligned} \tag{4.35}$$

where M is the feature size, ϵ is the privacy parameter, and $S_1 \triangleq \Sigma_{n \in C_1} x_{jn}^2 + \hat{x}_j^2$, $S_2 \triangleq \Sigma_{n \in C_2} x_{jn}^2$.

The bound in theorem 3 is quadratic in $\sqrt{t}$, and it generally allows a large tree number.

The variance of the class label estimation is preserved according to the following theorem:

Theorem 4 (Privacy Preservation of Class labels [38]) *The class label estimate $\tilde{y}$ preserves the class label variance if the number of trees, t, is bounded by*

$$t \leq \frac{1}{\pi(1 - \pi)}(N_l^+(N_l^+ + 1) + \frac{1}{\epsilon^2}) \tag{4.36}$$

where π is the probability of y belonging to class C_1, ϵ is the privacy parameter for the trees and N_l is the minimum leaf size.

4.5 Random forest algorithm

The approach in some of the articles was to focus on the structure of the forest algorithm. This could be done by *e.g.* improving the underlying trees to be more robust to noise, adjusting the forest parameters, or in some cases, pruning the decision trees.

4.5.1 Parameters

Majority of the DPRF articles use the standard procedure of building the random forest itself. This process includes determining the forest parameters, most commonly the maximum tree depth d_m and the tree amount t, and then building the forest accordingly [15].

Already in early work, Jagannathan et al. [21] recognize that the tree count and the tree depth are essential elements in DPRFs. They reference work by Fan et al. [12], who observe that in random tree ensembles, even a low tree count of 10 gives good results, and that a good choice for the tree depth is $d_m = m/2$, m denoting the amount of attributes in the data. In the experiments of [21], amounts as low as five trees resulted in acceptable accuracy, but the accuracy variance was higher compared to larger ensembles. The results also indicated that the depth $d_m = m/2$ does not always give the best results.

In [15] another way of determining the an optimal tree depth is proposed. The authors present an extension of the proposed tree depth, $m/2$, that also takes continuous attributes into account. The main issue with $d_m = m/2$ is that it assumes discrete attributes, which can only be selected once per root-to-leaf path. With s continuous attributes, the new approach starts with defining X as the expected number of untested continuous attributes, on a root-to-leaf path of depth d as:

$$E[X|d] = s(\frac{s-1}{s})^d. \tag{4.37}$$

The optimal tree depth is then proven to be $d = \arg\min_{d:X<s/2} E[X|d]$. The results of the article proved that the prediction accuracy of the algorithm improved with the new proposed depth compared to $d_m = m/2$ and $d_m = m$, where m is the number of features.

The tree parameters can also be chosen based on other information, like the size of the data or the privacy budget. In [14], the tree count t and a minimum support threshold θ are used as parameters, and both are fine-tuned based on the dataset size $|D|$, domain sizes of attributes $a \in A$, and the total privacy budget B. The number of trees t is first set to be the number of attributes $|A|$, $t = |A|$. In case there are many attributes, and therefore many trees, the privacy budget may spread too thin for adequate accuracy results. The value of θ is affected by the privacy budget, $\theta = (\sqrt{2}|C|)/\epsilon$ (where $|C|$ is the amount of unique class labels, and $\epsilon = B/t$), so dividing the budget to many trees can lead to too high values of the minimum support threshold θ (4.38). The tree count is reduced from $|A|$ until the constraint for θ (4.38) applies. The authors define a minimally acceptable tree size to be $d = 3$. For the minimally acceptable tree size to be viable, the minimum support threshold must comply with

$$\theta = \frac{\sqrt{2}|C|}{\epsilon} < \frac{|D|}{\delta^2}. \tag{4.38}$$

The δ corresponds to the average attribute domain size in the attribute set A. The tree count is defined as the largest value $t \leq |A|$, satisfying (4.38). In other words, the upper bound of t ensures that ϵ has large enough values for the minimally acceptable tree size, 3 [16].

The maximum depth is also not defined as a parameter in [38]. Instead, a parameter called minimum node support, which refers to the minimum leaf node size n_l, is required.

Like maximum depth, it is used as a termination criteria. The n_l is set as a small fraction of the total number of records in the data, and a node will not be split if any child node has a number of data points under n_l.

The optimal forest parameters may also be chosen based on conducted experiments with the algorithm. For example in [15], a so called *sweetspot* for the tree amount varied in size depending on the privacy budget ϵ. For example the window size for $\epsilon = 0.2$ was between 30 to 100 trees for good accuracy, increasing even up to 300 trees as the ϵ increased. Based on their observations and experiments, the authors recommend constructing 100 trees with their version of the DPRF.

In [6], the authors argue that a logarithmic size for the forest, $t = O(log(|D|))$, is sufficient in achieving good performance. Their DPRF algorithm uses random decision trees, with Laplace noise added to the leaf node statistics, and only binary decision trees are considered. The class labels in the training datasets are also assumed binary. The trees are complete, and of a fixed depth d_m. Three different techniques for determining the classification of a new data point are experimented with in the decision trees, and compared to one another: majority voting, threshold averaging and probabilistic averaging. Majority voting has the trees voting for the classification of a data point d with their respective predictions. Threshold averaging first calculates a fraction θ^d by summing the fractions of training points in a leaf l with label 1 across a set L of all leaves containing data point d, and then dividing this sum with t. The data point is predicted as class 1 if and only if $\theta^d > 0.5$. Probabilistic averaging also calculates θ^d, but the classification to class 1 happens with the probability of θ^d instead.

The authors present preliminary theorems on finding the optimal amount of trees for good accuracy. These theorems propose probabilistic bounds to the empirical error and generalization error. The empirical error is defined as the fraction of misclassified training data, and generalization error as the fraction of misclassified test data. The presented theorems for finding the best tree count based on the bounds are related to the three different data point classification methods. It should be noted that [6] does not provide proofs for the theorems.

Table 4.1 provides a summary of the forest parameters used in the articles. The table includes the best value for the tree depth and the tree count if reported in the article, and otherwise the formula used to calculate the parameter.

4.5.2 Decision trees

Sometimes, the decision tree algorithm itself is modified to be more resilient against the added noise, or otherwise more adjusted to the constraints of differential privacy. In [30] a new type of strategy to build the trees in the random forest is used, called a two-phase

Article	**Tree type**	d_m	t
Jagannathan et al. [21]	Random	m/2	10
Fletcher and Islam (2015) [14]	Random	-	$t \leq m$
Bojarski et al. [6]	Random	14	$O(log(\|D\|))$
Fletcher and Islam (2017) [15]	Random	$d = \underset{d:X<s/2}{\arg\min}\, E[X\|d]$	100
Consul and Williamson [7]	Median	8	10
Patil and Singh [37]	Greedy	5	20
Rana et al. [38]	Greedy	-	See theorems (4.35) and (4.36)
Li et al. [28]	Greedy	-	-
Hou et al. [20]	Greedy	7	50
Guan et al. [19]	Greedy	m/2	50
Zhang et al. [45]	Greedy	-	50
Liu et al. [30]	Greedy	5	90

Table 4.1: Random forest parameters d_m and t by article. Notation m refers to the attribute amount in A. Hyphen − indicates that the value was reported in the paper, nor any means to calculate it. The results are either the highest value of the parameter used in the experiments, the best found value of the parameter, or the formula used to calculate the value. Li et al. [28] report neither of the parameters, but they compare the results mostly with the Patil and Singh [37] DPRF. This makes it likely that the parameters used were similar with [37].

differential privacy random forest (TpDPRF). The tree building has two phases: first some decision trees are built with their corresponding bootstrapped training datasets, and then the rest of the trees are trained focusing on the samples that are intractable for the first set of trees. To accomplish this, the algorithm uses sample weighting. This technique assigns weights to the data samples based on their misclassification rate in the first tree-building phase.

The inputs of the forest are the training dataset D, the tree count in the forest B, the ratio of the first phase α determining the amount of trees that are required to be built in the first phase, and the total privacy budget, denoted with π. All the sample weights are first set to $1/|D|$. The amount of $\lfloor \alpha B \rfloor$ training datasets are subsequently created with bootstrap sampling, with the initial weights as input to the bootstrap sampling method. Finally, a set of differentially private decision trees is built on these datasets.

In the second phase, the sample weights are updated with rule $w_i = v_i / \Sigma_{i=1}^{n} v_i$, v_i being the misclassification count for sample x_i of the first phase. Then B - $\lfloor \alpha B \rfloor$ training datasets are created, with the updated weights as the input for the bootstrap sampling method, and the rest of the trees are trained. The pseudocode of TpDPRF is shown in Algorithm 4.1. Proofs for the decision tree algorithm, denoted as DP_Tree in Algorithm 4.1, achieving ϵ-differential privacy and the random forest algorithm achieving π-differential privacy are provided in [30].

Input: The original training data set $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, the size of random forest B, the ratio of first phase α, the total privacy budget π;

Output: A two-phase differential privacy random forest model *DPForest*;

1: **function** TPDPRF(D, B, α, π)
2: $DPForest = \{\}$;
3: $\epsilon = \frac{\pi}{B}$;
4: Initial weights for all samples $\omega_1 = \omega_2 = ... = \omega_n = \frac{1}{n}$;
5: **for** $i \in \{1, 2, ..., \lfloor \alpha B \rfloor\}$ **do**
6: D_i = sample (D, ω, n);
7: DP_TREE(D_i, F, d_{max}, d_T, C, η, γ, ϵ);
8: $DPForest = DPForest \cup Tree_i$;
9: Update the weights for all samples;
10: **for** $i \in \{\lfloor \alpha B \rfloor + 1, ..., B\}$ **do**
11: D_i = sample (D, ω, n);
12: DP_TREE(D_i, F, d_{max}, d_T, C, η, γ, ϵ);
13: $DPForest = DPForest \cup Tree_i$;
14: Return *DPForest*;

Algorithm 4.1: two-phase DPRF (TpDPRF), pseudocode. Image from Liu et al. [30] .

In [19] a method called selective aggregation is used alongside the random forest algorithm. The method aims to improve the accuracy of the forest by removing trees that do not promote the tree diversity in the ensemble. Ensemble learning theory states that differentiation between the learners, in this case the decision trees, improves the performance of the ensemble [19]. Following this theory, one decision tree (denoted as L_i) is removed and the remaining trees are aggregated to an ensemble, and to test if the removed decision tree should be kept or not, the performance of the ensemble is tested before and after the removal. If the error rate for the ensemble without L_i is minimal, removing L_i reduces decision tree similarity in the forest.

To optimize the time performance of the selective aggregation algorithm (Algorithm 4.2), the aggregation method is designed as iterative. The idea is to split the set of trees to multiple random subsets and perform the selective aggregation algorithm on them separately. Afterwards, the results are combined to a new base learner. The described steps are repeated until a termination criterion is met.

Algorithm 2: Selective Aggregation.

Input: Collection of τ trees ST_0
Output: Set of aggregated trees
for $i = 1$ *to* $(\tau = 1)$ **do**
  for $j = 1$ *to* $(\tau + 1 - i)$ **do**
    Remove the j-th basic learner L_{ij} from ST_{i-1}, where L_{ij} represents the j-th basic learner in the i-th round;
    Aggregate the remaining basic learners, and calculate the error rate e_{ij};
    Add the basic learner L_{ij} to ST_{i-1};
  $e_{ix} = \min\left\{e_{ij}, j = 1, 2, \cdots, (\tau + 1 - i)\right\}$;
  Generate a new basic learner set ST_i by removing the x-th basic learner;
  Calculate the error rate of ST_i and record as E_i;
$E_k = \min\{E_i, i = 1, 2, \cdots, (\tau - 1)\}$;
Return ST_k
end

Algorithm 4.2: Selective aggregation ensemble algorithm. Image from Guan et al. [19]

In [14], a new kind of label prediction and voting strategy is introduced, that is applicable to increase prediction accuracy when a DPRF uses random decision trees. The authors observed that the prediction accuracy rose if the node with the highest confidence was used to predict the class label. Because the used decision trees in [14] are random, a parent node can have a higher confidence compared to its leaves. This strategy is not useful in greedy trees, that commonly pick an attribute to split on that makes the child node's confidence better than the parents'. After building and pruning the forest, the most confident node is found in each existing root-to-leaf path, and then the prediction of this node is used for all future records that end up obeying this path. This is repeated

for a future record for each tree in the forest. In the voting phase, the highest confidence class value is picked as the final prediction result.

4.5.3 Pruning

As mentioned in the introduction to random forests, it is not strictly necessary to prune the decision trees in this type of an ensemble [41]. However, in some of the DPRF articles, pruning is used regardless. For example, the DPRF algorithm in [14] uses the signal-to-noise ratio SNR (equation 4.27) to prune nodes with SNR < 1, using *signal averaging* for non-leaf nodes.

Signal averaging aims to increase the strength of the signal μ compared to the obscuring noise σ [14]. Let X be the signal set, parameter μ the signal mean, and σ the standard deviation of the noise. If the probability of the noise decreasing or increasing the signal is the same, adding various signal measurements together produces a result with less noise:

$$SNR = \frac{\mu}{\sigma} = \frac{\sum_x^X \mu_x}{\sqrt{|X|\sigma^2}}, \tag{4.39}$$

The authors in [14] give critique to some earlier approaches of pruning differentially private trees that simply remove leaves that contain no records. Since the added noise to the leaf node class counts can also add noise to the zero counts (making them positive), all empty nodes are not necessarily removed from the trees with this pruning strategy. This can leave leaf nodes in the tree that have class counts constructed only of noise.

One of the termination criteria in [14] requires that when the estimated support of a node is low enough that the noise outweighs the signal (SNR < 1), the tree needs to stop growing. In order for SNR to be above 1, the estimated support of a node must be larger than the minimum support threshold θ (defined in equation 4.38). During node splitting, a child is created for each value v of the chosen splitting attribute a, and a splitting attribute a can only be picked once in a root-to-leaf chain. A node's estimated support is calculated by first assuming that the subset of records in the node D_i is divided equally amongst all the child nodes. Beginning from the root node, the support for each node is estimated by dividing $|D_i|$ with the domain size of the parent node's splitting attribute. Pruning is used after the forest construction to correct these rough estimates.

After the DPRF has been constructed, one query to each leaf-node will be made using the Laplace Mechanism to learn their class counts [14]. After the noisy class counts of each leaf are received, the nodes with acceptable SNR are re-checked. Because the class counts (with noise) are now known through the query, the earlier estimated supports can now be discarded: the class count of a parent node can be deduced by summing the class

counts of all its children. After these class counts are calculated for all the nodes, the SNR < 1 pruning rule is implemented. The formula (4.39) of SNR can be re-defined as

$$SNR = \frac{\epsilon \sum_{L}^{Leaves}(\sum_{c}^{C} L_c)}{|C|\sqrt{2 \times |Leafs|}}. \tag{4.40}$$

In the equation, L_c is the $c \in C$ class count of leaf L, and "Leaves" corresponds to the leaf node set reachable down from the current node.

4.6 Performance of existing DPRFs

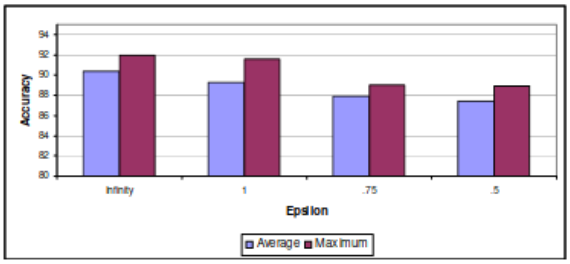
4.6.1 Random and non-greedy decision trees

The following results were obtained with DPRF implementations in the articles [21, 14, 6, 7] that did not use greedy heuristics in their splitting functions. The median splitting method used in [7] is not generally used as a splitting function in greedy decision trees due to low quality performance, so although it could technically be categorized as greedy, it is not referred to as such in the article. Due to most of the articles reporting their accuracy results in graphs, the exact accuracies can only be estimated from these images. In this thesis, the presented graphs and tables in figures 4.2 to 4.12 do not all depict the comprehensive reported results of their respective articles, but they are instead tactically limited to the most comparable UCI [32] datasets between the articles. The results presented in [15] are excluded due to the lack of proof of applying smooth sensitivity to the exponential mechanism as a replacement of global sensitivity.

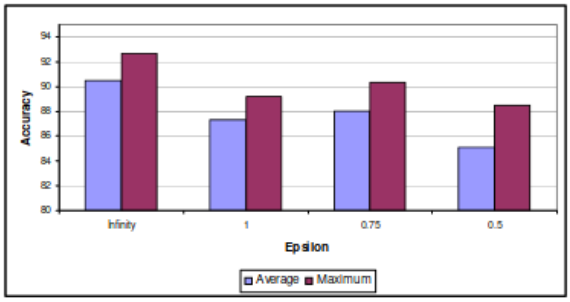
The articles also occasionally use different names of the same UCI datasets. To clarify these, the abbreviation WBC in Figure 4.3 refers to the Wisconsin-Breast Cancer UCI dataset, also referenced in figure 4.9 with the name Wisconsin. Also, datasets Car and Vehicle from figure 4.3 and 4.9 respectively, refer to the same UCI dataset.

Jagannathan et al. [21] used random decision trees in their DPRF algorithm, called Private-RDT. From the UCI datasets, they used Mushroom, Nursery and Congressional voting records. From the presented graphs in Figure 4.2, the average accuracy of the algorithm looks to have varied from 84% to 97%. The relatively poor accuracy results of the Congressional Voting Records data can be partly explained by the small dataset size (435 rows).

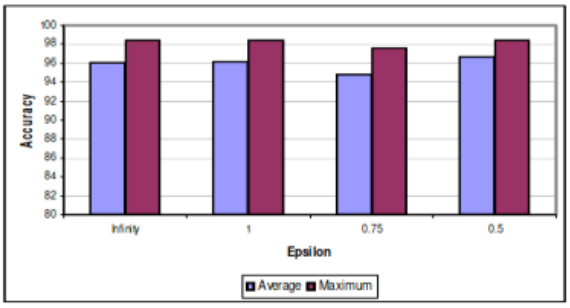
Fletcher and Islam (2015) [14] tested their DPRF algorithm, called DP-RF, with 9 datasets from the UCI Machine Learning Repository. The algorithm used random decision trees, and the highest confidence nodes to make predictions. The performance of DP-RF was compared to Private-RDT by Jagannathan et al. [21]. The results are presented in Figure 4.3. DP-RF performed better compared to Private-RDT on average



Accuracy on the Nursery data set from the UCI Repository. Displayed are the average and maximum accuracy for $\epsilon \in \{0.5, 0.75, 1, \infty\}$.



Accuracy on the Congressional Voting Records data set from the UCI Repository.



Accuracy on the Mushroom data set from the UCI Repository.

Figure 4.2: Jagannathan et al. [21] Private-RDT algorithm, experiments on accuracy with different values of the privacy budget. Image modified from the source.

in the achieved prediction accuracies.

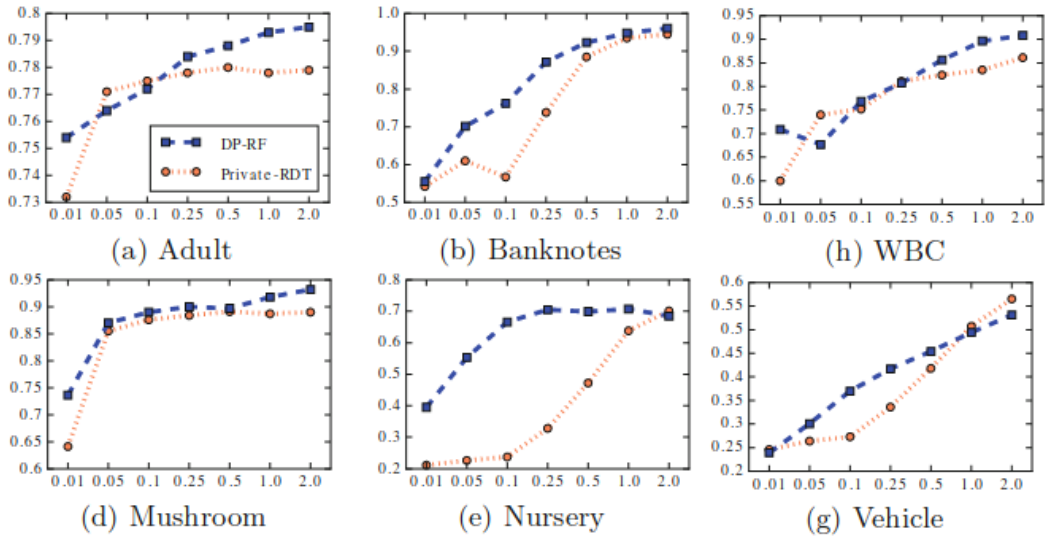


Figure 4.3: Fletcher and Islam (2015)[14] algorithm, experiments on accuracy with different values of the privacy budget. The y-axis represents the accuracy and x-axis the privacy budget. Image modified from source [14].

Bojarski et al. [6], used Banknote Authentication, Congressional voting records, Mammographic mass, and Mushroom UCI datasets. In figure 4.4 presenting the results,

dpRFMV corresponds to the differentially private random forest with majority voting, dpRFTA with threshold averaging and dpRFPA with probabilistic averaging. The results are reported using the test error (%) instead of accuracy. It can be seen from figure 4.4 that dpRFMV and dpRFTA outperform dfRFPA.

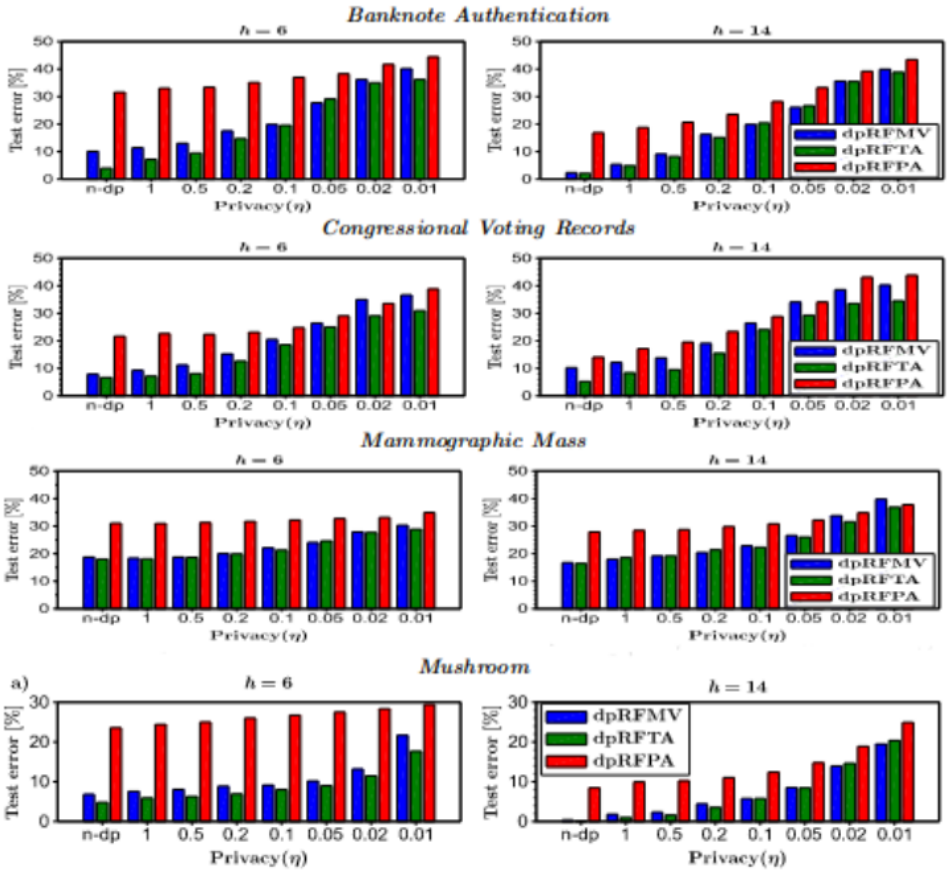


Figure 4.4: Test error results on four UCI datasets, and tree heights h=6 and h=14 from the algorithm in [6]. Image modified from Bojarski et al. [6]

Consul and Williamson [7] use three UCI datasets (Banknote Authentication, Credit card default, Wall-following robot navigation) in their experiments with classification. The DiPriMe algorithm [7] is compared with two existing DPRFs: DP-DF (by [13]) and DP-RF (by Patil and Singh [37]), and privatized extremely random trees, DP-ERT. While DiPriMe has clear accuracy loss compared to non-privatized algorithms for small budgets, it begins performing better as the budget increases, and it outperforms the other DPRF implementations, as can be seen in figure 4.5.

By comparing the results by the datasets in figure 4.3, DP-RF by [14] looks to outperform Private-RDT [21]. On low privacy budget values, DP-RF also looks to outperform DiPriMe by [7], comparing the results on the Banknotes UCI datasets: from figure 4.5, it can be seen that with $\epsilon = 0.1$, the accuracy of DiPriMe is around 60%, whereas DP-RF in figure 4.3 with the same privacy budget value achieves approximately 75% accuracy. The

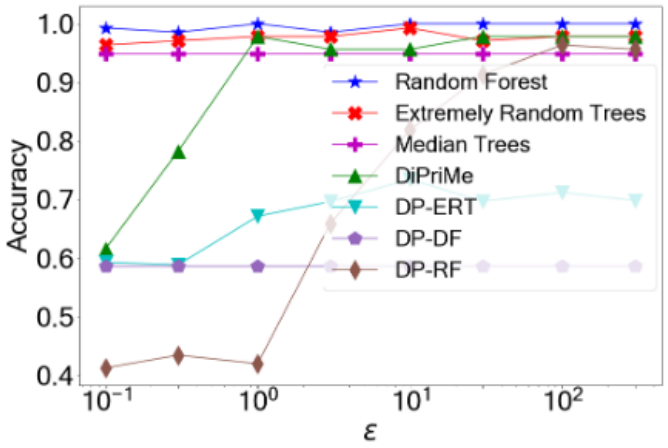


Figure 4.5: Accuracy performance based on the UCI banknote authentication data. Image from Consul and Williamson [7]

results of the Bojarski et al. [6] DPRF algorithms (dpRFMV, dpRFTA and dfRFPA) are reported by the test error instead of accuracy. By comparing figure 4.4 to the DP-RF results in 4.3, DP-RF and both dpRFMV and dpRFTA seem to achieve very similar results on Mushroom and Banknote datasets.

4.6.2 Greedy decision trees

Rana et al. [38], experimented with a relaxed definition of DP by only obscuring the variance of the necessary statistics with noise instead of the whole data. They test their algorithm on two datasets, one from UCI called Credit screening, and one non-UCI dataset, called Cancer. The Cancer data contains patient data from a regional hospital from Australia. In figure 4.6, that displays the results, the first row of graphs is from credit screening data, the second row is from the Cancer data. Area under the ROC curve (AUC) is used as the evaluation measure. RForest refers to the used DPRF, and DTree to a differentially private decision tree. The figure 4.6 illustrates how random forests overcome singular decision trees, and how in most cases increasing the forest size also improves the accuracy of the algorithm.

Hou et al. [20], implemented a novel privacy budget allocation scheme and a hybrid decision tree model that uses a new splitting function, called IG_GR. The algorithm was evaluated on two UCI datasets, Adult and Mushroom. When testing the algorithm's performance with both datasets, the results were compared to a standard non-private random forest (RF), and two other algorithms, DiffPRF (by Patil and Singh [37]) and DiffP-RFs (by Yin et al. [44]). The results are presented in figure 4.7. The algorithms performed better on the Mushroom dataset because it only has discrete attributes. On the Adult dataset, since DPRF by [20] uses a part of the privacy budget for the selection of

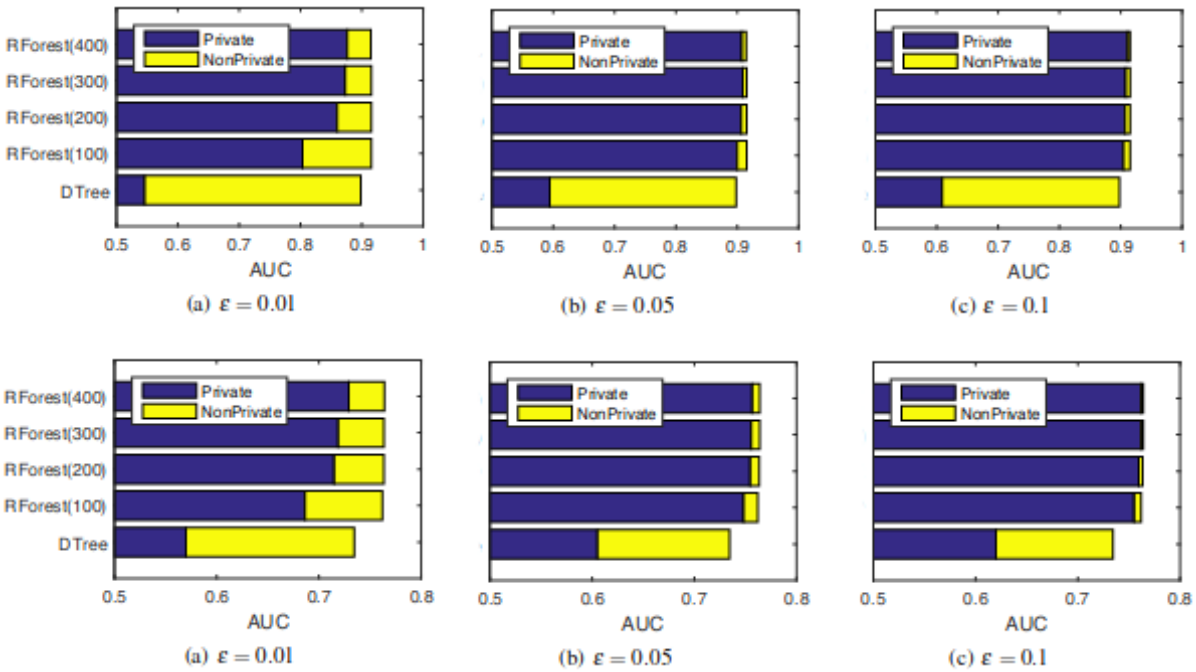


Figure 4.6: AUC results for the algorithm in [38] for two different datasets (first row: Credit Card Screening UCI, second row Cancer data). It should be noted that this algorithm is not DP, but uses a relaxed definition of differential privacy. Image modified from Rana et al. [38]

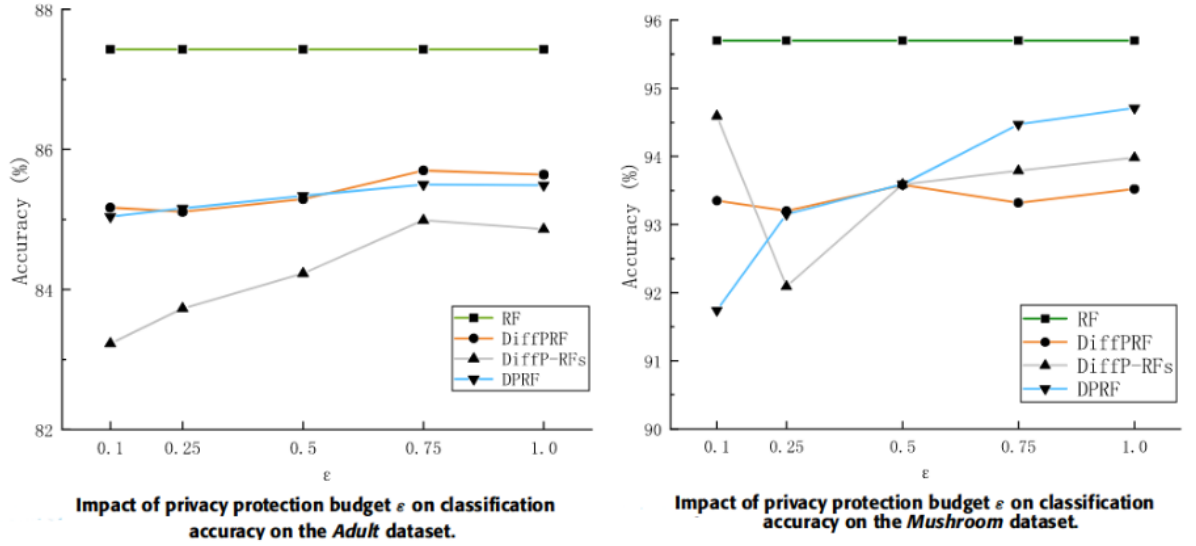


Figure 4.7: Hou et al. [20] DPRF algorithm performance on Adult and Mushroom UCI datasets. The results of the DPRF algorithm by [20] is marked with the blue line.

continuous attribute splitting points, the classification accuracy is slightly lower compared to DiffPRF, but it performs better than DiffP-RFs.

The DPRF introduced by Zhang et al. [45], called DPFMaxTree, used a novel attribute measurement method F_max and a new privacy budget allocation mechanism. DPFMaxTree is tested with the UCI datasets Adult and Mushroom. The accuracy of

DPFMaxTree is compared to two other DPRFs, MaxTree and the algorithm by Hou et al. [20], denoted in figure 4.8 graph as DPRF. In the figure, Max and FMax are non-differentially private random forests. It should be noted that Zhang et al. [45] report only using two attributes in the training datasets in their experiments. As can be seen from figure 4.8, DPRMaxTree outperforms DPRF on the Mushroom dataset, but is in turn worse when $\epsilon < 0.5$ on the Adult dataset. The amount of trees was also experimented with in contrast to accuracy. The best performance was achieved with 50 trees for most of the compared algorithms.

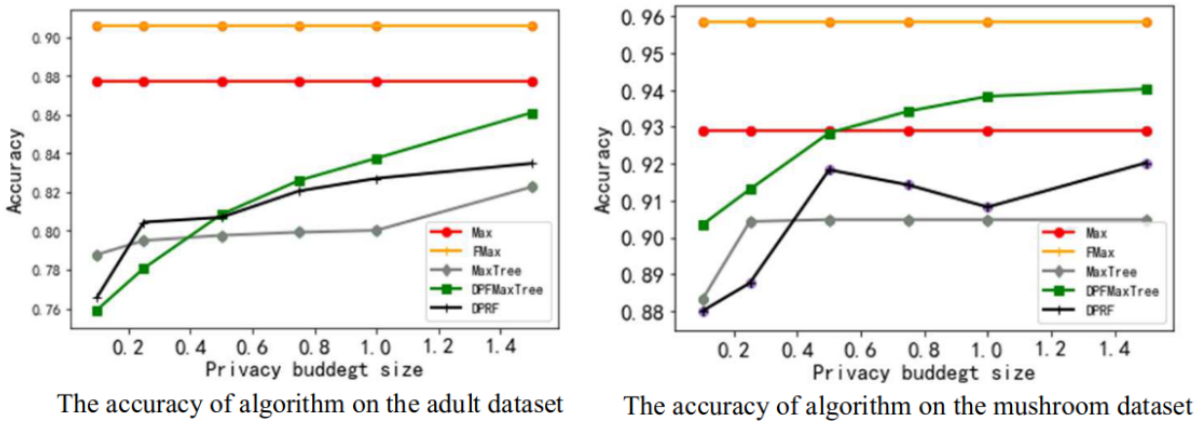


Figure 4.8: Accuracy comparisons with Adult and Mushroom UCI datasets. The results of DPRMaxTree are marked with the green line. Image from Zhang et al. [45].

Liu et al. [30], who created a two-phase random forest, evaluate their algorithm, TpDPRF, with six different datasets from the UCI: Wisconsin, Pima, Mammographic mass, Tic-Tac-Toe Endgame, Hungarian and Car. Figure 4.9 showcases the results of varying the privacy budget values contrasted to accuracy on three of these datasets. In the experiments, 30 trees and a maximum depth of 8 were used for the parameters. TpgDPRF is compared to three other DPRFs. The one denoted as DPRF in figure 4.9 is by Patil and Singh [37] and SNR is by Flecher and Islam [14]. The accuracy performance of TpDPRF seems to depend on the dataset.

Guan et al. [19] used a novel privacy budget allocation scheme and selective aggregation for the decision trees, and test the performance of their DPGDF algorithm with two datasets from UCI: the Adult dataset, and Heart disease dataset. The accuracy is compared with privacy-preserving and a baseline normal random forest. The presented accuracy of the algorithms is the average result of 10 iterations of 10-fold cross-validation, with recommended parameters. Figure 4.10 shows the comparison between the DPGDF and private random forest algorithms, denoted as DPDT (by [13]) and FS (FS is by Fletcher and Islam [15]). For the Adult dataset, DPGDF has an advantage in prediction accuracy under varying privacy budget. For the Heart disease dataset, the prediction

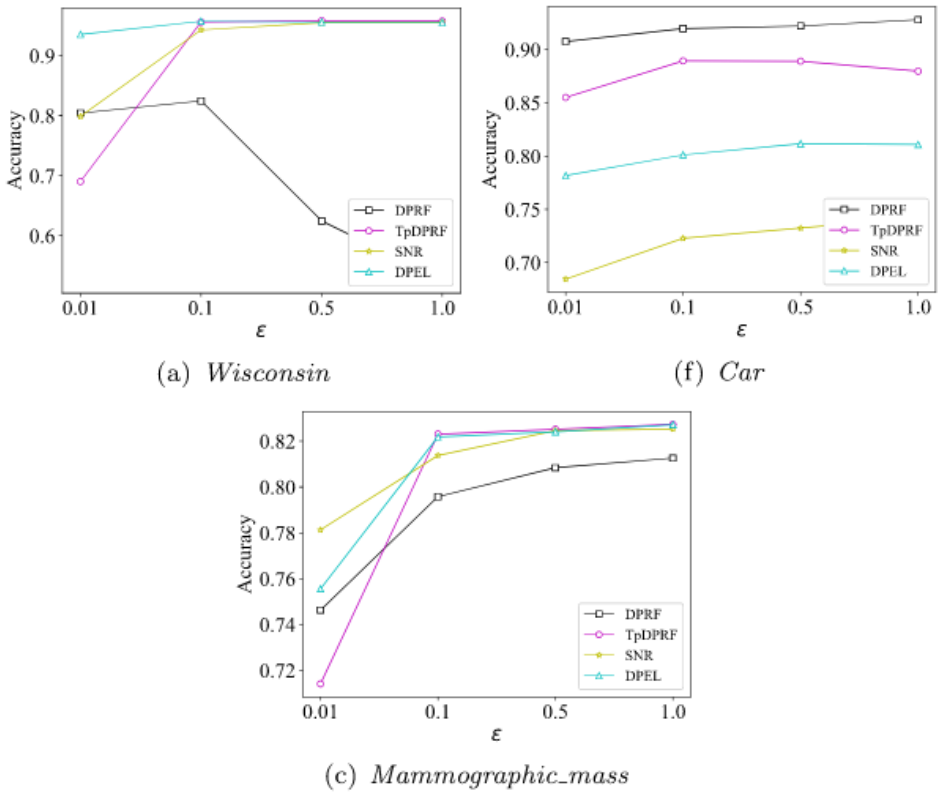


Figure 4.9: Liu et al. [30] TpDPRF performance. Image modified from source.

results are not on par with the results on the Adult data, since the dataset is small.

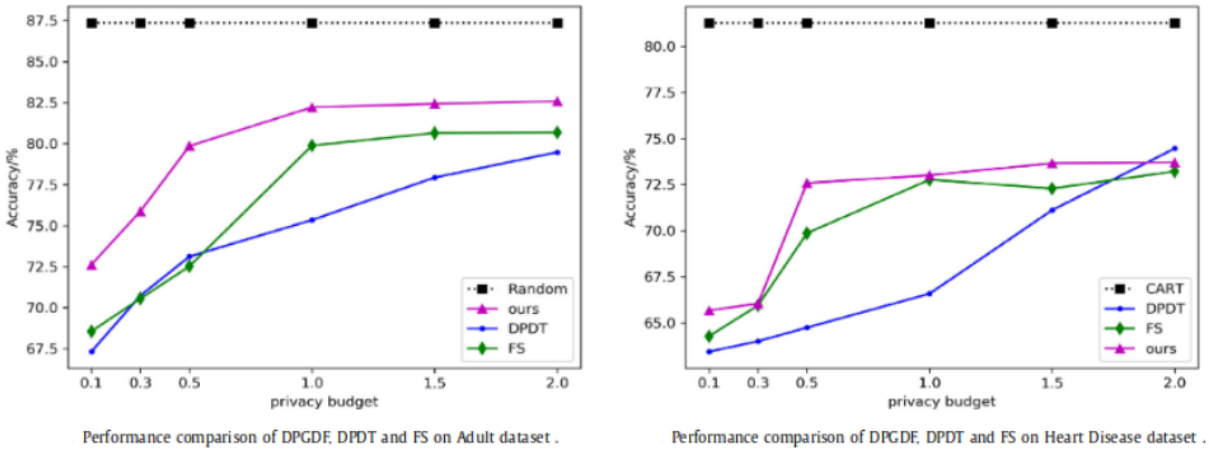


Figure 4.10: Accuracy results for the Adult and Heart Disease UCI datasets. Image modified from source Guan et al. [19]

Patil and Singh [37] experimented with different quality functions with their greedy random forest algorithm using ID3 decision trees. The split quality functions tested are the information gain, max operator and the Gini index. The experiments were run with different values of the privacy budget, and two different datasets from UCI, Mushroom and Adult. The information gain is sensitive to noise giving the lowest accuracies, and

max operator gives higher accuracies, as it is less sensitive. However, the differences in accuracy between the three splitting functions are not large, as can be seen from Figure 4.11.

CLASSIFICATION ACCURACY OF DIFFERENTIAL PRIVATE RANDOM FOREST FOR MUSHROOM DATASET WITH DIFFERENT QUALITY FUNCTIONS

Privacy budget(ϵ')	DiffPRF-InfoGain(%)	DiffPRF-Max(%)	DiffPRF-Gini(%)
0.1	93.35±1	93.89±1	93.34±1
0.25	92.04±1	93.82±2	93.58±2
0.5	93.58±1	94.07±1	94.19±1
0.75	93.32±1	93.82±1	93.58±1
1.00	93.48±1	93.14±1	93.51±1

CLASSIFICATION ACCURACY OF DIFFERENTIAL PRIVATE RANDOM FOREST FOR ADULT DATASET WITH DIFFERENT QUALITY FUNCTIONS

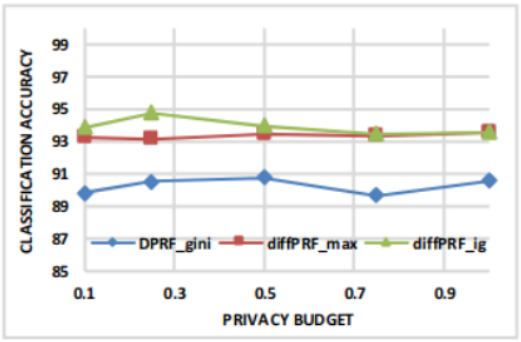
Privacy budget(ϵ')	DiffPRF-InfoGain(%)	DiffPRF-Max(%)	DiffPRF-Gini(%)
0.1	84.71±1	85.13±1	85.12±1
0.25	85.11±1	85.07±1	85.13±1
0.5	85.29±1	85.84±1	85.97±1
0.75	85.70±1	86.31±1	85.94±1
1.00	85.64±1	86.31±1	85.92±1

Figure 4.11: Performance comparison of quality functions. Tables from Patil and Singh [37]

The DPRF-Gini algorithm [28] was tested on the UCI datasets Car, Mushroom, Credit and Iris. The accuracy results are lower compared to the results of the other two algorithms, diffRF_max and diffPRF_ig, by Patil and Singh [37]. Results are shown in figure 4.12.

CLASSIFICATION ACCURACY OF DIFFERENT DATASETS

Dataset	*DiffPRF-IG(%)*	*DiffPRF-Max(%)*	*DPRF-Gini(%)*
Car	92.04±1	89.95±2	82.28±1
Mushroom	93.58±1	93.14±1	92.01±1
Credit	88.99±2	88.55±3	80.62±2
Iris	100.00±0	100.00±0	100.00±0



Classification accuracy of different privacy budgets

Figure 4.12: Table and graph from Li et al. [28] In the table, results are obtained with ϵ=0.1. The results presented in the graph are from the Mushroom dataset.

To summarize, the selective aggregation method of Guan et al. [19] DPGDF algorithm looks to improve on the results obtained by Flethcher and Islam [15]. The DPGDF algorithm still seems to perform worse in the Adult and Mushroom datasets compared to Zhan et al. [45] and Hou et al. [20] DPRFs. Out of the greedy tree random forests that used the strict definition of differential privacy, the algorithm of Hou et al. [20] in figure 4.7 seems to heed the best results, and the DiffPRF algorithm by Patil and Singh [37] has good accuracy results as well.

4.7 Summary and observations

The articles focus on various different aspects of the random forest algorithm to make it accustomed to the constraints of differential privacy. Since the forest algorithm's utility is compromised by the noise added to the data queries, strategies to alleviate these negative effects are the central topics. From privacy budget allocation to novel splitting functions, many elements are at play with each other, but certain groupings of the used methods are possible to identify.

The most obvious is the division between random and greedy decision trees. In the greedy decision trees of [20, 45, 30, 19], the approach to a privacy budget allocation scheme is that more sophistication is required for good accuracy results. By comparing the results of these sophisticated budget allocation strategy algorithms (Figures 4.7, 4.8, 4.9, 4.10, from [20, 45, 30, 19] respectively) and the two non-sophisticated greedy decision tree algorithms (Figures 4.11 [37] - 4.12 [28]), the accuracy results do not show obvious differences in low privacy budget values. Interestingly, the accuracy results between random tree and greedy tree ensembles, compared by *e.g.* Adult and Mushroom datasets, also do not seem to be too notable by comparing the figures. It could be that accuracy is still most closely linked with the dataset itself.

The balancing act of optimizing the forest parameters, such as the tree depth and the tree count, and the flow of data instances in the decision tree nodes in order to avoid situations where noise in the deeper nodes completely obscures the node statistics, was also a theme in the articles. Different tree amounts, from small numbers like 5 in [21] to larger values, like 90 in [30] were proposed. Similarly different suggestions appeared for tree depths, but for example in the experiments in [6] it can be seen that higher tree depths tend to result in decreased accuracy.

Some of the articles stood out from the others, like Liu et al. [30], who introduced a more complex random forest decision tree scheme, were the only ones to investigate feature selection, and Rana et al. [38], who used a relaxation of the differential privacy constraint, arguing that protecting attribute and class variances is enough. The difference in approach between applying bagging or sampling without replacement to the training data also divided the articles. When replacement is not used, like for example in [19], according to the parallel composition, the full budget can be used for all the trees since the data subsets are disjoint.

Overall, many different parts of the DPRF algorithm were investigated to tackle the privacy-utility problem inherent to differentially private algorithms. This chapter has provided an overview of these approaches based on twelve articles. An implementation of the DPRF algorithm will be tested in the next chapter, inspired by these previous works.

5. Implementation

Based on the review of previous works on DPRFs, an implementation of the DPRF algorithm is constructed, called $DPRF_{thesis}$. This algorithm will be compared to one similar and one different DPRF implementation based on its performance measured on accuracy and the privacy budget.

5.1 The DPRF

The three main properties of $DPRF_{thesis}$, inspired by the previous work, are

1. sampling without replacement,
2. using a privatized median to split internal nodes to child nodes,
3. using the exponential mechanism to make the predictions in leaf-nodes.

Inspired by Guan et al. [19], each decision tree is built on a random subset of the original training data, without replacement (i.e., disjoint subsets of the training dataset). Since the subsets are disjoint, according to the parallel composition theorem each tree can get the full privacy budget. The benefit of this strategy compared to sampling with replacement besides the larger privacy budget is that it promotes diversity amongst the trees, and reduces the variance caused by individual records [15].

Some of the articles discussed the benefits of using a non-sophisticated splitting function, or a splitting function that would not perform well in a non-differentially private setting, like the median splitting used by Consul and Williamson [7]. Random decision trees also tend to out-perform greedy trees according to the survey of Fletcher and Islam (2020) [16]. Inspired by these observations, the node splitting strategy of $DPRF_{thesis}$ is not designed based on traditional greedy heuristics in decision trees, and instead the nodes are split using differentially privatized medians, as designed by Consul and Williamson [7]. The privatized median splitting method has the benefit of avoiding low data occupancy in leaf nodes. For example, it is argued to overcome the random node splitting strategy of random decision trees, since even though the entire privacy budget can be used on the leaf node data queries, the amount of data ending on the leaf nodes will have high variance, leading to low-occupancy leaf-nodes that output less accurate results.

To summarize the node splitting strategy used by $DPRF_{thesis}$, the split point of an

attribute will be determined using the exponential mechanism, and the scoring function u is different depending on whether the attribute is continuous or discrete. For continuous attributes, the values of the attribute are from a set range of $[a_L, a_U]$. Ten splitting point values r are calculated from range $[a_L, a_U]$ as an ascending order list, with equal distance between adjacent values. The scores for the splitting points are then calculated as the difference of the number of data points X in node i falling under r, and over or equal to r. The exponential mechanism with this scoring function selects a splitting point with the following probability [7]:

$$Pr(r) \propto \exp(\frac{\epsilon_s}{2}||X_{i,a} \cap [a_L, r)| - |X_{i,a} \cap [r, a_U]||). \tag{5.1}$$

Similarly, the scoring function for the discrete attributes takes the absolute value of the size difference of attribute values belonging to category C and not belonging to category C. A splitting point for discrete attributes is chosen by the probability [7]:

$$Pr(C, X_i \backslash C) \propto \exp(\frac{\epsilon_s}{2}||C| - |X_i \backslash C||). \tag{5.2}$$

The splitting attribute is also chosen using the exponential mechanism. The negative mean squared error (MSE) is used as the scoring function. Its sensitivity is $4P^2/N_i$, where the target value lies in $[-P, P]$. P is set to be 1, since the target is a binary value. A splitting attribute is picked with the probability [7]:

$$Pr(\hat{a}) \propto \exp(\frac{\epsilon_a N_i}{8P^2}\text{MSE}(\hat{a})). \tag{5.3}$$

The exponential mechanisms were implemented in the code using IBM's Diffpriblib library. In (5.1), (5.2) and (5.3), the privacy budgets ϵ_a and ϵ_s are set to be equal to one another. The conducted splits at a depth d are done with disjoint data subsets, leading to a total privacy cost of $d_m(\epsilon_a + \epsilon_s)$ for the tree construction process. As in [7], all the internal node queries are given $\epsilon_a = \epsilon_s = (pB)/(2d_m)$ of the budget, where B is the full privacy budget and p is a probability value in range $p \in [0, 1]$. The value of p is set to 0.5. When the budget for leaf-node privatization, ϵ_l, is included, the total spent budget is $B = d_m(\epsilon_a + \epsilon_s) + \epsilon_l$.

The leaf-node queries receive $\epsilon_l = pB = B/2$ of the privacy budget. The mechanism used to determine the leaf-node classification value is the report one-sided noisy argmax (ROSNAM) mechanism, which is a special case of the exponential mechanism [10].

The ROSNAM exponential mechanism is designed for problems where the best of two possible classes needs to be selected based on which one is more common than the other [10]. If the class labels are A and B, and the true class counts are $n_A = 0$ and $n_B = f_c$, where $f_c > 0$, the utility u can be tied to the real class counts, and the sensitivity of the utility is $\Delta u = 1$. The class A utility is therefore 0, and the class B

utility f_c. Also, the maximum probability of an incorrect output (class A) is $2e^{-f_c*\epsilon/2}$. The ROSNAM mechanism adds noise to the utility of potential outputs drawn from a one-sided exponential distribution, with parameter $\epsilon/\Delta u$ in the case of a monotonic utility, and then reports the resulting argmax value. The utility is monotonic, since for a function to be monotonic the addition of an element to the dataset it uses as input does not cause decrease in the output values. This applies to class counts.

Since the method of determining the classification value of a leaf-node using the exponential mechanism simply outputs a privatized estimate of the most frequent class in the node, the ROSNAM exponential mechanism is suited for the task [15]. The addition of a record can only ever increase the utility, the utility of a given class c is defined as its class count frequency $u = f_c$. To demonstrate, with $\epsilon = 0.2$ and class frequencies $f_A = 6$ and $f_B = 4$, the probabilities are $Pr(A) \propto \exp(0.2*6) = 3.32$ and $Pr(B) \propto \exp(0.2*4) = 2.22$.

The ROSNAM exponential mechanism was implemented in the code using IBM's Diffpriblib library. The Permute-and-Flip mechanism [33] of the library is used. Since the class counts are monotonic, the mechanism provides $\frac{\epsilon}{2}$-differential privacy.

The tree depth for the Mushroom and Adult dataset is determined by the rule $d_m = m/2$, as in [21, 19]. The depth will be set at $d_m = 4$ for the Banknotes dataset, since this value achieved the best performance. The amount of trees t is also decided based on the performance of the algorithm. Since the samples to the decision trees are picked randomly without replacement, this creates a limit to the possible amount of trees, especially with smaller datasets. Still, a larger amount of trees has the benefit of *e.g.* averaging out the perturbation added by the exponential mechanism [15].

In $DPRF_{thesis}$, majority voting is used to determine the final classification result of a new record. The termination criteria, in addition to the maximum tree depth, are when the node data instances have only one unique class label, when the child node would have no data or less than 4 records, and when $|A| = 0$ would apply to the child node or when all the attribute values in the node data match each other. The size of the attribute subset for feature selection, which is always set to be smaller than $|A|$, was also decided based on the algorithm performance.

5.2 Data

The $DPRF_{thesis}$ algorithm can process both categorical and continuous attributes. The tests only include datasets for binary classification problems. Three UCI [32] datasets are used: Adult, Banknotes and Mushroom. Table 5.1 shows an overview of the used datasets.

The Adult dataset contains personal data details, like education level and marital status, to predict whether an individual earns more or less than 50,000$ per year. The

Dataset	Rows	Continous features	Discrete features	d_m
Adult	30162	5	8	7
Mushroom	5644	0	22	11
Banknotes	1372	4	0	4

Table 5.1: UCI dataset background information.

data contains categorical, ordinal and numerical input variables. The missing values were removed from the dataset in the tests, leaving 30162 rows. The data has a total of 14 attributes in addition to the target.

The Banknotes dataset consists of image data from genuine and forged banknotes. Four continuous features from the images are given to predict whether a banknote is forged or not. The data, with 1372 rows, does not contain missing values. The features were preprocessed by standardizing them by removing the mean and scaling them to unit variance.

The Mushroom dataset contains details about mushroom samples, like cap shape and habitat, to predict whether the mushroom is poisonous or not. There are 8124 samples in total of 23 different species. There are 22 different predictor input variables, all of them categorical. The rows with missing values were removed, leaving 5644 rows.

5.3 Performance

The tested values of the privacy budget are $\epsilon = \{0.01, 0.1, 0.25, 0.5, 0.75, 1, 2\}$. The results on the Banknotes, Mushroom and Adult UCI datasets are shown in figures 5.1, 5.2 and 5.3. 75% of the datasets was used as the training data, the rest as test data. The best results of the algorithm are represented with the green line. The results depicted with the blue line, are an average of 10 runs.

The accuracy variance between runs seems to be correlated with the privacy budget: for the Banknotes and Mushroom datasets, the smaller the privacy budget, the larger the accuracy differences between the runs of the algorithm. This can be seen in figures (5.1) and (5.2), where the distance between the average result and the best result generally increases as the budget decreases. Interestingly, the opposite applies to the Adult dataset (figure 5.3), where the differences in accuracy between runs on low privacy budget values were more similar than with higher budget values.

The most similar DPRF algorithm to $DPRF_{thesis}$ is DiPriMe random forest by Consul and Williamson [7]. DiPriMe uses privatized median to split the internal nodes, and the Laplace mechanism to determine the leaf-node class counts. The accuracy of DiPriMe on the banknote UCI dataset are shown in figure 4.5. A different DPRF algo-

rithm compared to $DPRF_{thesis}$ is presented by Hou et al. [20], which will be referred to as DPRF-Hou. This DPRF uses greedy heuristics with a novel splitting function IG_GR, and a novel budget allocation scheme. The results of DPRF-Hou are shown in figure 4.7.

Comparing the results of DiPriMe algorithm in figure (4.5) and $DPRF_{thesis}$ in figure (5.1), the accuracy goes through a similar drop with small ϵ values, dropping to around 60% in $\epsilon = 0.01$ with DiPriMe and to 56% with $DPRF_{thesis}$, but starting to reach higher accuracies when $\epsilon \geq 1$. The results of DiPriMe are comparable to the best achieved results of $DPRF_{thesis}$, but the average accuracy results of the $DPRF_{thesis}$ do not achieve over 90% accuracy even with high privacy budget values, but reach around 88% accuracy above $\epsilon \geq 0.25$. This could be due to the different leaf-node privacy mechanisms, as DiPriMe uses Laplace noise to obscure the true class counts.

$DPRF_{thesis}$ and DPRF-Hou by [20] are compared based on the Adult and Mushroom UCI datasets. Based on the accuracy results of DPRF-Hou shown in figure 4.7 and $DPRF_{thesis}$ results in figure 5.3, on the Adult dataset DPRF-Hou clearly outperforms $DPRF_{thesis}$, consistently achieving around 85% accuracy for all tested values of the privacy budget. The best accuracy achieved by $DPRF_{thesis}$ on the Adult dataset is around 81% on $\epsilon = 1$, and the accuracy drops to 74% on the lowest privacy budget values. On the other hand comparing the results on the Mushroom dataset in figure 4.7 and figure 5.2, $DPRF_{thesis}$ outperforms DPRF-Hou: on $\epsilon = 0.1$, $DPRF_{thesis}$ achieves around 94% average accuracy, and DPRF-Hou slightly below 92% accuracy. Also on a higher privacy budget value, $\epsilon = 1$, DPRF-Hou achieves around 95% accuracy, whereas the average accuracy achieved by $DPRF_{thesis}$ is around 98%.

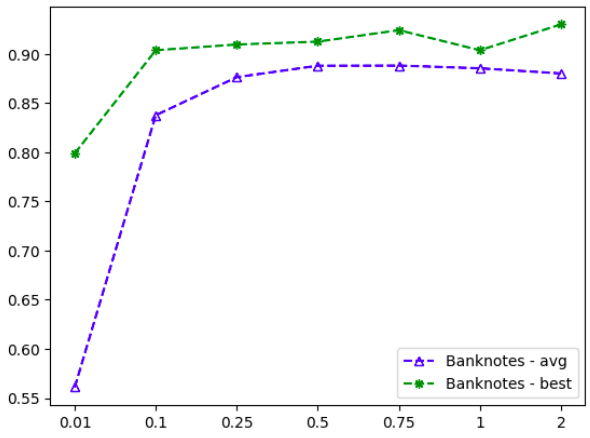


Figure 5.1: Accuracy results of the $DPRF_{thesis}$ algorithm on the Banknotes UCI dataset. The average accuracy is calculated from 10 runs. Maximum depth $d_m = 4$, tree count $t = 9$.

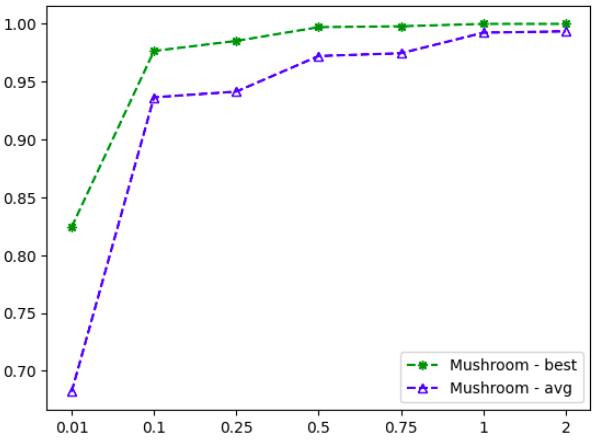


Figure 5.2: Accuracy results of the $DPRF_{thesis}$ algorithm on the Mushroom UCI dataset. The average accuracy is calculated from 10 runs. Maximum depth $d_m = 11$, tree count $t = 6$.

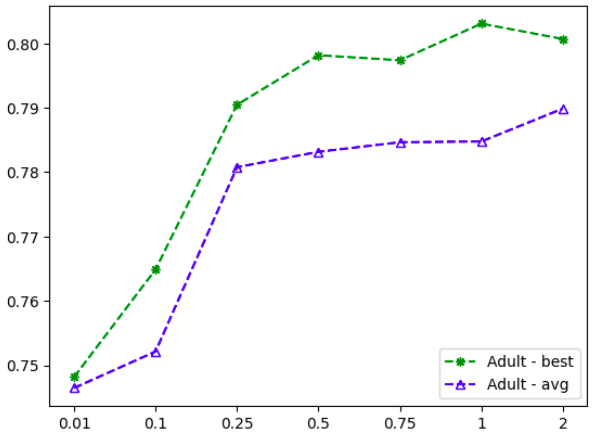


Figure 5.3: Accuracy results of the $DPRF_{thesis}$ algorithm on the Adult UCI dataset. The average accuracy is calculated from 10 runs. Maximum depth $d_m = 7$, tree count $t = 15$.

6. Conclusion

This thesis has explored differential privacy applied to random forest classification algorithms. This was done by first introducing the preliminary concepts of differential privacy and decision tree ensembles, and then reviewing twelve articles about existing implementations of the algorithm. Based on the articles, an implementation of the algorithm, $DPRF_{thesis}$, was constructed, and compared to two other DPRF algorithms. The algorithm used privatized medians to split the internal nodes of the decision tree, which was the approach taken by Consul and Williamson [7]. The main goal of using median splits was to reduce the amount of low-occupancy nodes in the decision trees. Determining the classification of a leaf-node requires the differentially privatized approximation of the most frequent class in the node. The ROSNAM exponential mechanism is used to determine the classification results of new records, which is the approach taken by Fletcher and Islam, in an addendum to their original article [15]. The forest sizes of $DPRF_{thesis}$ algorithm were decided based on the dataset size and the best accuracy performance, and the maximum depths of the decision trees were decided based on performance and the attribute sizes of the used dataset. The privacy budget was distributed to the data queries with the same approach as in [7].

The accuracy results of figures 5.1, 5.2 and 5.3 are reported as averages of 10 runs, and the best achieved accuracies of all runs. The accuracy results followed the standard logic of differential privacy applied to algorithms: the smaller the privacy budget, the less accurate the results, and vice versa. The performance of the $DPRF_{thesis}$ algorithm was tested on three UCI datasets, and its performance is compared to a similar algorithm DiPriMe [7], and less similar algorithm DPRF-Hou [20]. The best results obtained with $DPRF_{thesis}$ were comparable to the performance of DiPriMe on the Banknotes dataset, but the average accuracy results of $DPRF_{thesis}$ were lower compared to DiPriMe. $DPRF_{thesis}$ outperformed DPRF-Hou on the Mushroom dataset, but the accuracy results achieved by DPRF-Hou on the Adult dataset were better compared to $DPRF_{thesis}$.

This thesis focused mostly on the effect of epsilon on the performance of the algorithm. In the future, the algorithm can be improved and tested more with different tree depths and tree counts. There is also some ambiguity with using mean squared error (MSE) as the scoring function for the median splits for determining the splitting attribute,

since it is not a standard method to use with classification tasks, and is instead mostly used in regression. A method more suited to classification could be tested in the future. The code of the algorithm is available at https://github.com/ssuihko/DPimplementation.

Bibliography

[1] *Classification and regression trees*. Wadsworth, Belmont (CA), 1984.

[2] E. Alpaydin and F. Bach. *Introduction to Machine Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, 3rd ed. edition, 2014.

[3] M. Ati. Big Data Security and Privacy Implementation: The way Ahead. In *2020 IEEE 7th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, pages 1–5, Kuala Lumpur, Malaysia, Dec. 2020. IEEE.

[4] A. Blanco-Justicia, D. Sanchez, J. Domingo-Ferrer, and K. Muralidhar. A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning. *ACM Computing Surveys*, 55(8):1–16, Aug. 2023.

[5] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, Baltimore Maryland, June 2005. ACM.

[6] M. Bojarski, A. Choromanska, K. Choromanski, and Y. LeCun. Differentially- and non-differentially-private random decision trees, Feb. 2015. arXiv:1410.6973 [cs].

[7] S. Consul and S. Williamson. Differentially Private Random Forests for Regression and Classification. *Association for the Advancement of Artificial Intelligence*, 2021.

[8] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. Mathematics for Machine Learning. page 412.

[9] C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[10] C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2013.

[11] H. Elaidi, Z. Benabbou, and H. Abbar. A comparative study of algorithms constructing decision trees: ID3 and C4.5. In *Proceedings of the International Conference on*

Learning and Optimization Algorithms: Theory and Applications, pages 1–5, Rabat Morocco, May 2018. ACM.

[12] W. Fan, H. Wang, P. Yu, and S. Ma. Is random model better? on its accuracy and efficiency. In *Third IEEE International Conference on Data Mining*, pages 51–58, 2003.

[13] S. Fletcher and M. Z. Islam. A differentially private decision forest. *AusDM*, 15:99–108, 2015.

[14] S. Fletcher and M. Z. Islam. A Differentially Private Random Decision Forest Using Reliable Signal-to-Noise Ratios. In B. Pfahringer and J. Renz, editors, *AI 2015: Advances in Artificial Intelligence*, volume 9457, pages 192–203. Springer International Publishing, Cham, 2015. Series Title: Lecture Notes in Computer Science.

[15] S. Fletcher and M. Z. Islam. Differentially Private Random Decision Forests using Smooth Sensitivity. *Expert Systems with Applications*, 78:16–31, July 2017. arXiv:1606.03572 [cs].

[16] S. Fletcher and M. Z. Islam. Decision Tree Classification with Differential Privacy: A Survey. *ACM Computing Surveys*, 52(4):1–33, July 2020.

[17] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in pharmacogenetics: An End-to-End case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, San Diego, CA, Aug. 2014. USENIX Association.

[18] C. Gentile and N. Littlestone. The robustness of the p -norm algorithms. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 1–11, Santa Cruz California USA, July 1999. ACM.

[19] Z. Guan, X. Sun, L. Shi, L. Wu, and X. Du. A differentially private greedy decision forest classification algorithm with high utility. *Computers & Security*, 96:101930, Sept. 2020.

[20] J. Hou, Q. Li, S. Meng, Z. Ni, Y. Chen, and Y. Liu. DPRF: A Differential Privacy Protection Random Forest. *IEEE Access*, 7:130707–130720, 2019. Conference Name: IEEE Access.

[21] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A Practical Differentially Private Random Decision Tree Classifier. In *2009 IEEE International Conference on Data Mining Workshops*, pages 114–121, Miami, FL, USA, Dec. 2009. IEEE.

[22] I. Jarin and B. Eshete. DP-UTIL: Comprehensive Utility Analysis of Differential Privacy in Machine Learning. In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy*, pages 41–52, Baltimore MD USA, Apr. 2022. ACM.

[23] B. Jayaraman and D. Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, Santa Clara, CA, Aug. 2019. USENIX Association.

[24] Z. Ji, Z. C. Lipton, and C. Elkan. Differential Privacy and Machine Learning: a Survey and Review, Dec. 2014. arXiv:1412.7584 [cs].

[25] O. Kotevska, F. Alamudun, and C. Stanley. Optimal Balance of Privacy and Utility with Differential Privacy Deep Learning Frameworks. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 425–430, Dec. 2021.

[26] A. Leroux, M. Boussard, and R. Des. Information gain ratio correction: Improving prediction with more balanced decision tree splits, 2018.

[27] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd international conference on data engineering*, pages 106–115. IEEE, 2006.

[28] Z. Li and S. Li. Random forest algorithm under differential privacy. In *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pages 1901–1905, Oct. 2017. ISSN: 2576-7828.

[29] X. Lin, C. Li, W. Ren, X. Luo, and Y. Qi. A new feature selection method based on symmetrical uncertainty and interaction gain. *Computational Biology and Chemistry*, 83:107149, 2019.

[30] J. Liu, X. Li, Q. Wei, S. Liu, Z. Liu, and J. Wang. A two-phase random forest with differential privacy. *Applied Intelligence*, Oct. 2022.

[31] Y. Lu, T. Ye, and J. Zheng. Decision Tree Algorithm in Machine Learning. In *2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pages 1014–1017, Aug. 2022.

[32] K. N. Markelle Kelly, Rachel Longjohn. The UCI machine learning repository, 2023.

[33] R. McKenna and D. Sheldon. Permute-and-Flip: A new mechanism for differentially private selection, Oct. 2020. arXiv:2010.12603 [cs].

[34] A. Narayanan and V. Shmatikov. How To Break Anonymity of the Netflix Prize Dataset, Nov. 2007. arXiv:cs/0610105.

[35] J. P. Near and C. Abuah. Programming Differential Privacy.

[36] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 75–84, San Diego California USA, June 2007. ACM.

[37] A. Patil and S. Singh. Differential private random forest. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2623–2630, Sept. 2014.

[38] S. Rana, S. K. Gupta, and S. Venkatesh. Differentially Private Random Forest with High Utility. In *2015 IEEE International Conference on Data Mining*, pages 955–960, Nov. 2015. ISSN: 1550-4786.

[39] S. Singh and P. Gupta. COMPARATIVE STUDY ID3, CART AND C4.5 DECISION TREE ALGORITHM: A SURVEY. *International Journal of Advanced Information Science and Technology*, 2014.

[40] L. Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[41] G. Williams. Random Forests. In *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery*, pages 245–268. Springer New York, New York, NY, 2011.

[42] R. Xu, N. Baracaldo, and J. Joshi. Privacy-Preserving Machine Learning: Methods, Challenges and Directions, Sept. 2021. arXiv:2108.04417 [cs].

[43] Y. Yang. Discretization. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 370–371. Springer US, Boston, MA, 2017.

[44] Y. Yin, L. Chen, Y. Xu, and J. Wan. Location-Aware Service Recommendation With Enhanced Probabilistic Matrix Factorization. *IEEE Access*, 6:62815–62825, 2018.

[45] Y. Zhang, P. Feng, and Y. Ning. Random Forest Algorithm Based on Differential Privacy Protection. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 1259–1264, Oct. 2021. ISSN: 2324-9013.

[46] J. Zhao, T. Wang, T. Bai, K.-Y. Lam, Z. Xu, S. Shi, X. Ren, X. Yang, Y. Liu, and H. Yu. Reviewing and Improving the Gaussian Mechanism for Differential Privacy, Dec. 2019. arXiv:1911.12060 [cs].

[47] T. Zhu, G. Li, W. Zhou, and P. S. Yu. *Differential Privacy and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2017.