A Differentially Private Decision Forest

Sam Fletcher **Md Zahidul Islam**

School of Computing and Mathematics,
Centre for Applied Machine Learning (CAML)
Charles Sturt University,
Bathurst, New South Wales 2795, Australia,
Email: `safletcher@csu.edu.au`
Email: `zislam@csu.edu.au`

Abstract

With the ubiquity of data collection in today's society, protecting each individual's privacy is a growing concern. Differential Privacy provides an enforceable definition of privacy that allows data owners to promise each individual that their presence in the dataset will be almost undetectable. Data Mining techniques are often used to discover knowledge in data, however these techniques are not differentially privacy by default. In this paper, we propose a differentially private decision forest algorithm that takes advantage of a novel theorem for the local sensitivity of the Gini Index. The Gini Index plays an important role in building a decision forest, and the sensitivity of it's equation dictates how much noise needs to be added to make the forest be differentially private. We prove that the Gini Index can have a substantially lower sensitivity than that used in previous work, leading to superior empirical results. We compare the prediction accuracy of our decision forest to not only previous work, but also to the popular Random Forest algorithm to demonstrate how close our differentially private algorithm can come to a completely non-private forest.

Keywords: Differential Privacy, Decision Forest, Data Mining, Machine Learning, Privacy.

1 Introduction

Data collection and analysis plays an ever-growing role at in all facets of society, whether it be economic, medical, political, militaristic, academic or anything in between. For some of these areas, people's privacy is a concern that needs addressing. This most often occurs when data is collected about individuals, with some of the data being "sensitive" – that is, data that an individual would not like becoming public knowledge. Regardless of individuals' motivations for a desire for privacy, it is considered a basic human right by many, including the U.N. (UN General Assembly 1948), and is codified in many country's laws. It is therefore vital that when a company, government agency, or any other party is performing data collection and analysis, they have methods for guaranteeing the privacy of those individuals whose data is being used. In many situations, individuals may simply refuse to offer their data for collection if they do not have a privacy guarantee.

Differential privacy (Dwork 2006, Dwork et al. 2006, Dwork 2007, 2008, 2011, Dwork & Roth 2014, McSherry & Talwar 2007, McSherry 2009) is one such method for guaranteeing individuals' privacy. It does so by making a promise to each individual who supplies information to a dataset: "Any information that could be discovered about you with your data in the dataset could also, with high probability, be discovered *without* your data in the dataset". In other words, the output of any query Q performed on dataset D will be indistinguishable from the output of the same query Q performed on dataset D', where D' differs from D by at most one record (the record of any individual). The privacy guarantees made by differential privacy are far greater than those made by other popular privacy-preservation methods, such as k-anonymity (Sweeney 2002, LeFevre et al. 2005) or other generalization or noise addition techniques (Fung et al. 2010, Fletcher & Islam 2015). We define differential privacy in full in Section 2.1.

If the output of Q is to be restricted in some way to enforce differential privacy, the next question is "How do we enforce differential privacy while still outputting useful knowledge?". It is this question that we will answer, by querying the dataset in a way that allows us to produce a high quality, differentially private Decision Forest. A Decision Forest is the term used for a collection of different Decision Trees, which are a common type of classifier used in Data Mining. Decision Trees work by iteratively selecting attributes in the dataset that can most accurately classify a "class attribute"[1]. When an attribute is selected, the records in the dataset are split up according to what value they have for the chosen attribute[2]. For each of these partitions, the process is then repeated until a termination condition is met. Common termination conditions include a maximum number of times a partition will be split, a minimum number of records remaining in a partition, or when a partition can classify the class attribute with 100% accuracy. Our proposed Decision Forest will be based off CART (Breiman et al. 1984), and is explained in more detail in Section 2.2.

1.1 Problem Statement

Dataset D is a two-dimensional matrix of rows and columns, where each row (i.e. record) $r \in D$ de-

[1]The class attribute is the attribute that the user wishes to accurately predict the value of for future records, where the value is not known.

[2]The process is slightly different for continuous attributes that lack discrete values, but we will be focusing on discrete (i.e. categorical) attributes in this paper. Note that methods exist to "discretize" continuous attributes (Kotsiantis & Kanellopoulos 2006).

scribes a single individual, and each column is an attribute a in the set of attributes A. Each r possesses one discrete value $v \in a; \forall a \in A$. We symbolize that record r has value v for attribute a by writing $r_a = v$. The subset $D_{a_v} \subseteq D$ is the subset in which $r_a = v; \forall r \in D_{a_v}$. A subset with multiple value requirements is denoted similarly, with each requirement separated by a comma (see Figure 1).

Each r also has a class value c, from the class attribute C. The aim of a decision forest is to correct predict r_C (the class value c of record r) for records $r \in B : B \cap D = \emptyset$, where B and D are drawn from the same population.

A user is given limited access to D, in which they are allowed to query D in an ϵ-differentially private way. For any given query Q, the value of ϵ can be equal to or less than the amount provided to the user by the data owner. We define this amount as the total privacy budget β, and will be dividing β into smaller parts for each query Q.

Our aim is to build τ decision trees ($0 < \tau < \infty$) by only submitting ϵ-differentially private queries Q to D, and without exceeding our total budget β. The decision trees should be of acceptably high quality so that meaningful knowledge can be discovered.

1.2 Our Contributions

Our contributions can be summarized as the following:

- We present a differentially-private decision forest algorithm (referred to as DPDF).
- We output all the rules in the forest, including their confidence and support.
  - We also output the confidence and support of every subset of the rules (i.e. increasingly more general versions of the final rules, by removing the antecedents one at a time).
- We propose and prove a theorem for the worst-case local sensitivity[3] of the Gini Index.
- We demonstrate that the prediction accuracy of our DPDF is close to the accuracy of a standard Random Forest classifier (Breiman 2001), even for low ϵ values.

We also provide a URL to our publicly available code for DPDF.[4] In Section 2 we provide some necessary background on differential privacy, the CART algorithm and the prediction accuracy measure. In Section 3 we propose our differentially private decision forest algorithm and discuss the important components of it. In Section 4 we discuss another technique that is similar to ours, DiffPID3 (Friedman & Schuster 2010), and how it differs from our technique. In Section 5 we empirically compare our technique to DiffPID3, using Random Forest (Breiman 2001) as a benchmark of prediction accuracy on various datasets. In Section 6 we conclude the paper.

2 Background

2.1 Differential Privacy

While differential privacy has many applications beyond those used in this paper (McSherry & Talwar 2007), and can be phrased more generally to encompass those applications, we phrase the below definitions in a way that is more specific to our scenario, with dataset D.

Definition 1 (Differential Privacy (Dwork 2006)). *A query $Q : Q(D) \rightarrow Y$ satisfies ϵ-differential privacy if for all datasets D and D' differing by at most one record,*

$$Pr(Q(D) = y \in Y) \leq e^{\epsilon} \times Pr(Q(D') = y \in Y) \ . \quad (1)$$

This definition allows a data collector to make a strong promise to each individual in D: that for any query Q, the output observed is $\frac{1}{\exp(\epsilon)}$ as likely to occur even if they had not been in D. It does not promise that a malicious user cannot find out any information about them, but it does promise that any information they can find, they could have found without the individual even being in D. For example, there might exist a strong pattern that the malicious user knows (by using secondary information outside D) an individual matches, and that pattern would exist in D with or without the individual.

In order for Definition 1 to be possible for query Q to achieve, there must be a randomized component in Q, preventing any output y from being 100% likely. Two mechanisms are commonly used to inject randomness into queries: the Laplace Mechanism and the Exponential Mechanism. Before we define these mechanisms, we first need to define the "sensitivity" of Q:

Definition 2 (Sensitivity (Dwork et al. 2006)). *A query Q has sensitivity $\Delta(Q)$, where:*

$$\Delta(Q) = \max_{K,K'} |Q(K) - Q(K')| \quad (2)$$

and K and K' are any datasets that differ by at most one record.

Using Definition 2, we now define:

Definition 3 (The Laplace Mechanism (Dwork et al. 2006, Dwork & Roth 2014)). *A query Q satisfies ϵ-differential privacy if it outputs $y + Lap(\frac{\Delta(Q)}{\epsilon})$, where $y \in Y : Q(D) \rightarrow Y$ and $Lap(x)$ is an i.i.d. random variable drawn from the Laplace distribution with mean 0 and scale x (i.e. variance $2x^2$).*

Note that the Laplace Mechanism requires $Y \rightarrow \mathbb{R}$. For non-real outputs, we can use the Exponential Mechanism:

Definition 4 (The Exponential Mechanism (McSherry & Talwar 2007)). *Using a utility function $u(Q, y) : u \rightarrow \mathbb{R}$ where u has a higher value for more preferable outputs $y \in Y$, a query Q satisfies ϵ-differential privacy if it outputs y with probability proportional to $\exp\left(\frac{\epsilon u(Q,y)}{2\Delta(u)}\right)$. That is,*

$$Pr(Q(D) = y) \propto \exp\left(\frac{\epsilon \times u(Q, y)}{2\Delta(u)}\right) \ . \quad (3)$$

We will later take advantage of two more theorems that have been proven about differential privacy:

Definition 5 (The Composition Theorem (McSherry & Talwar 2007)). *The application of queries Q_i, each satisfying ϵ_i-differential privacy, satisfies $\sum_i \epsilon_i$-differential privacy.*

Definition 6 (The Parallel Composition Theorem (McSherry 2009)). *Let D_i be a disjoint subset of dataset D. Let $Q_i(D_i)$ satisfy ϵ-differential privacy; then $\sum_i Q_i(D_i)$ also satisfies ϵ-differential privacy.*

[3]Sensitivity is an important component of Differential Privacy, described later.

[4]Our code can be found at `http://csusap.csu.edu.au/~zislam/` or you can email us.

2.2 CART

The structure of our decision trees will follow a similar structure to CART (Breiman et al. 1984). CART uses a recursive process in which attributes A (describing the records in D) are used to classify class attribute C. For any given attribute $a \in A$, the records in D are split into disjoint subsets D_{a_v} defined by which value $v \in a$ they have. We define each D_{a_v}'s ability to correctly classify a record's class value $c \in C$ using the Gini Index (Breiman et al. 1984):

$$G_{D,C}(a) = -\sum_{v \in a} \frac{|D_{a_v}|}{|D|} \left(1 - \sum_{c \in C} \left(\frac{|D_{a_v,c}|}{|D_{a_v}|} \right)^2 \right) \ . \tag{4}$$

The Gini Index is a measure of how often a randomly chosen record $r \in D$ would be incorrectly predicted to have class value $c : c \neq r_C$ if the predicted class value was randomly drawn from the distribution of class labels in D.

The attribute a with maximum $G_{D,C}(a)$ is chosen to split D into disjoint subsets $D_{a_v}; \forall v \in a$, and the process is repeated with each D_{a_v} being considered as it's own dataset D. The recursive process terminates when one of the following conditions are met:

- All records in D have the same class value c.
- All attributes have been used previously in the current recursion chain (an attribute can only be used once, since records in D_{a_v} cannot be further split by a).
- $|D_{a_v}|$ is below a user-defined minimum size. This is often done to prevent over-fitting to the training data.
- The depth of D_{a_v} in the tree has reached a user-defined maximum (i.e. the current recursion chain has repeated the maximum number of times). This is often done to limit computational complexity and limit the complexity (length) of the decision rules (the number of attributes required to predict the class attribute).

The output of this recursive algorithm is a decision tree T. An example of a decision tree can be seen in Figure 1. A tree T can be considered as a graph, and in this context (and no longer in the context of the recursive algorithm) the subsets D_i are often called "nodes", where i represents all the attributes in A (and the values of those attributes) that were used to define the subset D_i. The first node (i.e. subset) D is known as the "root node" (at $d = 1$ in Figure 1), and the final nodes D_i are known as "leaf nodes" (at $d > 1$ in Figure 1 if no more nodes exist lower in the chain).

2.3 Prediction Accuracy

The success of a decision tree T made with CART is usually measured with the prediction accuracy measure, and we use this measure in our paper. Prediction accuracy works by taking each record $r \in B : r \notin D; \forall r \in B$ (i.e. records not used in the tree-building process[5]) and following the logic of T into the disjoint subset D_{a_v} that matches the value r_{a_v}, starting from the first attribute used to split D (called the "root node") until the final split at the end of the recursion chain (called the "leaf node"). Note that it is

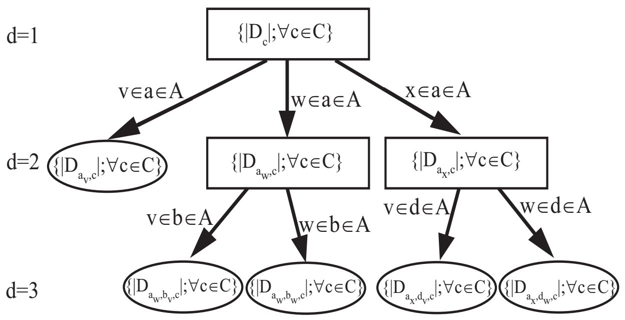

Figure 1: An example of a decision tree. Each arrow lists the value of an attribute that the records are being filtered according to. D is a dataset, with D_i being a subset of the dataset meeting the criteria i; C is the set of class values; A is the set of attributes, with each attribute having a set of values; d is the depth of the tree.

only possible for r to match one $root \rightarrow leaf$ recursion chain in T. We then predict that $r_C = c$, where c is the most common class value in the leaf that r matched. After repeating this for all $r \in B$, the percentage of cases where our prediction was correct is called the "prediction accuracy" of T.

To calculate prediction accuracy for a forest F, the predicted class value for a record r from each tree $T \in F$ is tallied, and an algorithm is used to select the most appropriate class value to predict overall. This is known as "voting", and various voting algorithms have been developed over the years (Breiman 2001, Islam & Giggins 2011).

3 A Differentially Private Decision Forest

In order to make a decision tree (and from there a decision forest), we need to consider what information we require about D. In order to make the decision tree differentially private, we then need to query D in a way that returns to us all the information we need, without distorting the answers too much.

We break down the tree-building process into the following queries:

$$P_1(D_i) = \{|D_{i,c}|; \forall c \in C\} \tag{5}$$

and

$$P_2(D_i) = a' \in A : G_{D_i,C}(a') = \max_{a \in A}(G_{D_i,C}(a)) \ , \tag{6}$$

where $D_i \subseteq D$, and i represents all the attributes in A (and the values of those attributes) that define what records are in D_i. These two queries are all that is required to recursively build the tree described in Section 2.2.

To make these queries differentially private, we add uncertainty to the output of each query in the following way:

$$Q_1(D_i) = \{|D_{i,c}| + Lap(\frac{1}{\epsilon}); \forall c \in C\} \tag{7}$$

and

$$\begin{aligned} Q_2(D_i) = a' \in A :& \\ Pr(G_{D_i,C}(a') &= \max_{a \in A}(G_{D_i,C}(a))) \\ &\propto \exp\left(\frac{\epsilon \times G_{D_i,C}(a)}{2\Delta(G)} \right) \ . \end{aligned} \tag{8}$$

[5] B is often called the "testing data", as opposed to the "training data" D used to "train" the tree about the patterns in D.

In words, $Q_1(D_i)$ outputs a histogram of the frequencies of each class value c in D_i, with Laplace noise added to each class count using Definition 3. The scale of the Laplace distribution is equal to $\frac{1}{\epsilon}$ for several reasons: when outputting a count of records, the sensitivity of the count is always $\Delta = 1$ (Dwork et al. 2006); and when outputting a histogram, the sensitivity remains equal to one because each bin of the histogram is disjoint − adding or removing one record can only affect one bin in the histogram (see Definition 6).

In words, $Q_2(D_i)$ outputs the attribute with the best Gini Index result with a certain probability, as described in Definition 4. The probability of any attribute $a \in A$ being outputted is dependent on the Gini Index of a, the sensitivity of the Gini Index, and ϵ. Since $\Delta(G)$ and ϵ are unchanging for all a, we can see that an attribute with a good Gini Index is exponentially more likely to be outputted than an attribute with a poor Gini Index. High $\Delta(G)$ and low ϵ both reduce this likelihood; we discuss how we reduce $\Delta(G)$ to a much lower value than that suggested by Definition 2 in Section 3.1.

The next question is how to divide the total privacy budget β amongst $Q_1(D_i)$ and $Q_2(D_i)$ for all i. Recall that a D_i exists for every $v \in a$, for every a used to split the previous D_i in the recursive algorithm described in Section 2.2, until each recursion chain has met one of the termination conditions listed in Section 2.2. Figure 1 provides an illustration of this. As per the Composition Theorem (Definition 5), we know that the ϵ we use for each query will be summed together, until it totals β and our connection to the dataset is severed by the data owner. We also know that for each depth level d, each D_i is disjoint, and $\sum_i |D_i| = |D|$ for each d (i.e. all records will belong to one and only one D_i for each d). We can therefore treat each D_i on the same depth level in a similar way to how we treat bins in a histogram, and that adding or removing one record can only affect one D_i. This allows us to apply the Parallel Composition Theorem (Definition 6) and ask Q_1 and Q_2 in a way that returns the output for each D_i, and only subtract ϵ from β once for Q_1 and once for Q_2.

Given the above, each query we use is given the following ϵ budget:

$$\epsilon = \frac{\beta}{2\delta - 1} \tag{9}$$

where $0 < \delta < \infty$ is the maximum depth we allow d to reach. We multiply δ by two because we are asking two queries per depth level: Q_1 and Q_2. We subtract one because upon reaching the end of a recursion chain (a leaf in the tree), we only need to ask Q_1 (to get the final distribution of class values).

We also expand our algorithm to be capable of producing multiple trees. Decision forests are known to often produce higher prediction accuracy than a single decision tree (Breiman 2001, Islam & Giggins 2011). The reason for this depends on the decision forest algorithm used, but is essentially because each tree will select different attributes a at different points in the tree, leading to different rules that might have better accuracy. For our algorithm, we guarantee that each tree will be different by requiring the first attribute chosen (the root) to be different for each tree. The noisy output of Q_2 also provides potential for different attributes to be chosen at other points in the trees.

When calculating the prediction accuracy of forest F in this paper, we use a simple voting technique of taking the weighted average of the predicted class values of record r, where the weight of each tree T's prediction is defined by the confidence[6] of the most common class value in the leaf that record r fits into. This is repeated for each record r in the testing data B. Any other voting technique can easily be used though, as long as it can be calculated using only the distribution of class values in D_i (unless some of budget β remains for more queries).

After defining the number of trees τ to build, the first time each tree (after the first tree) asks Q_2, the attributes chosen as roots of the previous trees are removed. There will therefore be τ different attributes used as the root attributes of the τ trees. The trees are completely independent beyond that.

We adjust the budget ϵ given to each query to account for τ:

$$\epsilon = \frac{\beta}{\tau(2\delta - 1)} \tag{10}$$

where $1 < \tau < \infty$.

We provide the full algorithm for DPDF in Algorithm 1.

3.1 The Local Sensitivity of the Gini Index

The sensitivity seen in Definition 2 is sometimes referred to as the "global sensitivity" of query Q, due to it making no assumptions about the data and simply returning the worst possible difference between any two datasets K and K' that differ by one record. By using the output of query $Q_1(D_i)$, we learn information about D_i that allows us to greatly reduce the sensitivity of $Q_2(D_i)$. We provide a theorem for the local sensitivity of $G_{D,C}(a)$ (and therefore of $Q_2(D_i)$), and its proof, below:

Theorem 1 (Local Sensitivity of the Gini Index)**.** *The sensitivity of the Gini Index $\Delta(G_{D,C}(a))$ when applied to data with known size $|D|$ is*

$$\Delta(G_{D,C}(a)) = 1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2 . \tag{11}$$

It is independent of a and C, and therefore we can abbreviate $\Delta(G_{D,C}(a))$ to $\Delta(G_D)$.

Proof. From Equation 2 we see that in order to maximize $\Delta(G_{D,C}(a))$ we must produce a maximum and a minimum $G_{D,C}(a)$ such that both outputs are possible by only changing one record in D. Using Equation 4, we reduce the problem to

$$\max \left| \sum_{c \in C} \left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2 - \sum_{c \in C} \left(\frac{|D'_{a_v,c}|}{|D'_{a_v}|}\right)^2 \right| \tag{12}$$

and will then extrapolate to all $v \in a$.

For D, we can write $\sum_{c \in C} \left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2$ in a more general way:

$$\sum_i \left(\frac{x_i}{y}\right)^2 \tag{13}$$

where $\sum_i x_i = y$. For D', if we assume that we are considering $v : r_a = v$, where r is in D' but

[6] "Confidence" is the percentage of records in D_i that have the most common class value.

not in D, and that $x_1 = c : r_C = c$ we can write $\sum_{c \in C} \left(\frac{|D'_{a_v,c}|}{|D'_{a_v}|}\right)^2$ as

$$\left(\frac{x_1+1}{y+1}\right)^2 + \sum_{i=2}\left(\frac{x_i}{y+1}\right)^2 . \tag{14}$$

That is, the class count x_1 and the total y were increased by one because r was added to D'. It is possible that $x_1 = 0$ for D, which we write as

$$\left(\frac{1}{y+1}\right)^2 + \sum_{i=2}\left(\frac{x_i}{y+1}\right)^2 . \tag{15}$$

Remembering that $\sum_i x_i = y$, we can separate the numerators and denominators of Equations 13, 14 and 15 and see that

$$\begin{aligned}(y+1)^2 - y^2 &> \left((x_1+1)^2 + \sum_{i=2}(x_i)^2\right) - \sum_{i=1}(x_i)^2 \\ &> \left(1 + \sum_{i=2}(x_i)^2\right) - \sum_{i=1}(x_i)^2 \end{aligned} \tag{16}$$

which means the denominator is guaranteed to increase by more than the numerator (and thus result in a smaller number) when adding r to D, and Equation 15 will always be smaller than Equation 14. Thus we maximize Equation 12 by using Equations 13 (for D) and 15 (for D').

By taking advantage of the fact that

$$\sum_{i=1}^{2}\left(\frac{x_i}{y}\right)^2 > \sum_{i=1}^{>2}\left(\frac{x_i}{y}\right)^2 \tag{17}$$

and

$$\left(\frac{y}{y}\right)^2 > \left(\frac{x_1}{y}\right)^2 + \left(\frac{x_2}{y}\right)^2 \tag{18}$$

where $\sum_i x_i = y$, we can see that the worst-case scenario (i.e. where Equation 12 is maximized) is when $|D_{a_v,c}| = |D_{a_v}|$ (i.e. all records in D_{a_v} have the same class value c) and $D' = D \cup r$ where $r_C \neq c$. That is,

$$\max\left(\sum_{c \in C}\left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2\right) = \left(\frac{|D_{a_v}|}{|D_{a_v}|}\right)^2 \tag{19}$$

and

$$\begin{aligned}&\min\left(\sum_{c \in C}\left(\frac{|D'_{a_v,c}|}{|D'_{a_v}|}\right)^2\right) \\ &= \left(\frac{|D_{a_v}|}{|D_{a_v}|+1}\right)^2 + \left(\frac{1}{|D_{a_v}|+1}\right)^2 , \end{aligned} \tag{20}$$

meaning that Equation 12 is equal to

$$\left|\left(\frac{|D_{a_v}|}{|D_{a_v}|}\right)^2 - \left(\left(\frac{|D_{a_v}|}{|D_{a_v}|+1}\right)^2 + \left(\frac{1}{|D_{a_v}|+1}\right)^2\right)\right| . \tag{21}$$

If Equation 21 is the maximum difference for $v \in a$, then it is also the maximum difference $\forall v \in a$, meaning the weighted average performed in the Gini Index (Equation 4) can be simplified:

$$\begin{aligned}&-\sum_{v \in a}\frac{|D_{a_v}|}{|D|}\left(1 - \sum_{c \in C}\left(\frac{|D_{a_v,c}|}{|D_{a_v}|}\right)^2\right) \\ &= -\left(1 - \sum_{c \in C}\left(\frac{|D_c|}{|D|}\right)^2\right) . \end{aligned} \tag{22}$$

From Equations 19 and 20, we know that Equation 12 (and therefore Equation 22) is optimal when all records in D have the same class value. Therefore when considering Equation 22 for D, we get

$$-\left(1 - \left(\frac{|D|}{|D|}\right)^2\right) = -(1-1) = 0 , \tag{23}$$

and when considering Equation 22 for D', we get

$$-\left(1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2\right) . \tag{24}$$

Combining the solutions for D and D' (Equations 23 and 24), we arrive at

$$\begin{aligned}&\Delta(G_{D,C}(a)) = \max_{D,D'}|G_{D,C}(a) - G_{D',C}(a)| \\ &= \left|0 - \left(1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2\right)\right| , \end{aligned} \tag{25}$$

noting that the above proof holds for the alternate case where one record is removed from a dataset by considering $|D| = |D'| - 1$.

□

Using Theorem 1 we can calculate the global sensitivity of the Gini Index, where the size of D is not known (recalling that the sensitivity is when $\Delta(G)$ is maximal):

$$\begin{aligned}\Delta(G) &= \max_{0<|D|<\infty}(1 - \left(\frac{|D|}{|D|+1}\right)^2 - \left(\frac{1}{|D|+1}\right)^2) \\ &= 1 - \left(\frac{1}{1+1}\right)^2 - \left(\frac{1}{1+1}\right)^2 \\ &= 0.5 .\end{aligned}$$

If $|D|$ is known, even a modest size of $|D| = 100$ heavily reduces the sensitivity of the Gini Index:

$$\begin{aligned}\Delta(G_D) &= 1 - \left(\frac{100}{100+1}\right)^2 - \left(\frac{1}{100+1}\right)^2 \\ &= 0.0196 .\end{aligned}$$

This drastically reduces the amount of noise added to the outputs of queries using the Exponential Mechanism (Definition 4) where the Gini Index (Equation 4) is the utility function u.

For DPDF, we can calculate $|D|$ using the sum of the class value frequencies we learned with query $Q_1(D)$:

$$|D| = \sum_{f \in Q_1(D)}|f| \tag{26}$$

Using this logic, we define a minimum number of records in D before the recursion chain is terminated (as explained in Section 2.2). For our experiments we set the minimum size of D to 100, thus making the upper limit of the sensitivity of Q_2 equal to $\Delta(G_D) = 0.0196$.

3.2 Pruning the Tree

Aside from calculating $|D_i|$, there is another advantage to using query $Q_1(D_i)$ at every D_i node, and not just in the leaf nodes where we need the class value distribution for predicting $r_C; \forall r \in B$ (where B is the testing data). By knowing the class distribution in every D_i node in every $root \rightarrow leaf$ recursion chain, it allows us to compare the leaf nodes to their parent nodes (the node above them in the chain) and assess their quality. By "quality", we mean "ability to correctly classify records r in B". If a parent node has higher quality than it's average child node (assuming that all the child nodes are leaf nodes), we perform what is known as "pruning".

Pruning is a component of many decision tree (and forest) algorithms, where leaf nodes are removed if they do not increase the prediction capabilities of the tree (Breiman et al. 1984, Quinlan 1993). The longer the $root \rightarrow leaf$ chains are, the more complicated they are, and the more they divide the records into smaller subsets, potentially over-fitting the tree to the training data D at the expense of prediction accuracy on the testing data B. If a leaf node is not actively helping the tree, it is more beneficial to remove the leaf. Some pruning techniques achieve pruning by using a validation set $B' : B' \cap B = \emptyset \ and \ B' \cap D = \emptyset$, such as CART's minimal cost complexity pruning (Breiman et al. 1984) and reduced error pruning (Quinlan 1993). However these techniques reduce the size of the training dataset D, which would increase the amount of relative noise added by Q_1 and Q_2. Since this is something we want to avoid, we instead perform pruning using the information we have already gathered during the tree-building process, specifically with Q_1.

Since all values $v \in a$ need to be handled, instead of removing individual bad leaf nodes, we measure the average quality of all leaf nodes $\forall v \in a$ and compare that to their parent node (which they all have in common). If

$$G(Q_1(D_i)) \geq \sum_{v \in a} \frac{|D_{i,a_v}|}{|D_i|} G(Q_1(D_{i,a_v})) \qquad (27)$$

where $D_{i,a_v} \forall v \in a$ are leaf nodes, we remove all of D_i's leaf nodes, causing D_i to become a leaf node instead. In the above equation, a is the attribute used to split D_i. Also recall that $Q_1(D_i)$ is the distribution of class values in D_i, which is the only information required to calculate the Gini Index G.

The reason that Equation 27 can possibly be true is that our tree building algorithm always splits a node D_i until a termination condition is met, even if all possible attributes to choose from have lower Gini Index results than D_i. Even if some attributes are better, the noisy output of $Q_2(D_i)$ might cause a poor attribute to be chosen. By allowing these situations to potentially occur, we help DPDF avoid getting stuck in a local optima – even if a child node y has worse Gini Index than it's parent node x, the child node z of y could still beat x's Gini Index! Including this pruning step in our algorithm then checks if either of these situations occurs, and retracts the tree to the global optima within the space explored by T.

3.3 Outputting the Rules and Subrules

Not only does possessing the output of $Q_1(D_i); \forall i$ allow us to perform pruning, but it also allows us to have many more rules than would be possible if we only had the class distribution of the leaf nodes. By a "rule", we mean the attributes chosen along a $root \rightarrow leaf$ chain, leading to the prediction of a certain class value c with a certain confidence. Depending on what sort of knowledge the user is searching for, these rules can be extremely valuable, allowing the user to see humanly-readable patterns in the data. The alternative is that tree T (or forest F) merely acts as a "black box" classifier, where records $r \in B$ are inputted and a predicted class value is outputted, with no information on *why*.

Not only does DPDF output the $root \rightarrow leaf$ rules, but also all $root \rightarrow node$ subsets of the $root \rightarrow leaf$ rules. This is only possible because we know what the predicted class value is for every node D_i, and the confidence of that prediction. Given that the user has a very strict privacy budget β with which to learn about dataset D, the more information we can gain from our queries – and the more we can recycle that information for multiple purposes – the better.

4 Related Work

Attempts at differentially private decision trees have been made in the past, most notably the Differentially Private ID3 algorithm (DiffPID3) by Friedman & Schuster (2010).

DiffPID3 uses a similar algorithm to our method, with some very important exceptions:

- Instead of getting the distribution of class values in each D_i, Q_1 just returns a noisy count of $|D_i|$.
  - This places limitations on DiffPID3's ability to prune the tree (Friedman & Schuster 2010).
  - It also does not allow for the first termination condition we use at each D_i subset (i.e. node): "All records in D_i have the same class value c", listed in Section 2.2. This increases the computation time of the algorithm by a small amount, both because it causes the algorithm to spend time making redundant nodes, and also because it increases the amount of pruning that needs to be done.
- They provide an extension to their algorithm to handle continuous attributes, however only at heavy cost to the budget β. Their results suggest a more feasible solution will need to be developed to handle continuous attributes with realistic β values.
- They do not discuss extending their algorithm beyond a single tree, while our algorithm builds τ distinct trees.
- They divide the privacy budget less efficiently, with $\epsilon = \frac{\beta}{2\delta}$, which makes a non-trivial difference at common values of δ.
- Most importantly of all, they use the global sensitivity of the Gini Index ($\Delta(G) = 0.5$), adding a huge amount of unnecessary noise to their tree and reducing the prediction accuracy of the tree, as seen in Section 5.

We demonstrate in Section 5 that our improvements over their algorithm have a substantial positive effect on the prediction accuracy of the classifier.

Algorithm 1 The proposed Differentially Private Decision Forest (DPDF)

```
procedure DPDF($D$, $C$, $A$, $\beta$, $\tau$, $\delta$)
    $\epsilon = \frac{\beta}{\tau(2\delta-1)}$
    $A_{root} = \{\}$                                  ▷ $A_{root}$ will be used as a global variable
    $F = \{\}$
    for $t = 1, \dots, \tau$ do
        $T$ = BuildTree($D$, $C$, $A$, $\epsilon$, $\delta$, 1, True)      ▷ The forest is composed of trees
        $T$ = PruneTree($T$)
        $F = F \cup T$
    end for
    return $F$
end procedure

procedure BuildTree($D$, $C$, $A$, $\epsilon$, $\delta$, $d$, root)
    T = {}                                             ▷ The start of a tree, or a subtree
    $N^C_D = \{|D_c| + Lap(\frac{1}{\epsilon}); \forall c \in C \in D\}$   ▷ i.e. $Q_1(D)$, the noisy frequency of each class value in $D$
    $|D| = \sum_{f_c \in N^C_D} f_c$
    if $d \leq \delta$ and $|D| \geq 100$ and $\frac{f_c}{|D|} < 1; \forall f_c \in N^C_D$ and $|A| > 0$ then
        $\Delta(G_D) = 1 - (\frac{|D|}{|D|+1})^2 - (\frac{1}{|D|+1})^2$     ▷ The worst-case local sensitivity
        $a$ = SplittingAttribute($D$, $C$, $A$, $\epsilon$, $\Delta(G_D)$, root)
        $A = A - \{a\}$                    ▷ We cannot split on an attribute twice in a $root \to leaf$ chain
        for all $v \in a$ do                           ▷ Recall that $A$ is public knowledge
            $D_{a_v} = \{r : r_a = v, \forall r \in D\}$   ▷ Note that $D_{a_v}$ must remain on the server to preserve privacy
            $T = T \cup$ BuildTree($D_{a_v}$, $C$, $A$, $\epsilon$, $\delta$, $d+1$, False)   ▷ Attach a subtree to the tree
        end for
    end if
    return $\{T, N^C_D\}$   ▷ The tree is composed of nodes, each composed of a subtree and a Class Histogram
end procedure

procedure SplittingAttribute($D$, $C$, $A$, $\epsilon$, $\Delta(G_D)$, root)
    if root then
        $A = A - A_{root}$                         ▷ Two trees cannot have the same root attribute
    end if
    $a'$ = Using the Exponential Mechanism, return $a' \in A$ :
            $Pr(G_{D_i,C}(a') = \max_{a \in A}(G_{D_i,C}(a))) \propto \exp\left(\frac{\epsilon \times G_{D_i,C}(a)}{2\Delta(G)}\right)$     ▷ i.e. $Q_2(D)$
    if root then
        $A_{root} = A_{root} \cup \{a'\}$                          ▷ Recall that $A_{root}$ is global
    end if
    return $a'$
end procedure

procedure PruneTree($T$)
    for all Parent Nodes $P \in T$ do               ▷ A Parent Node is any node with Child Nodes
        Child Nodes $H^P$ = {All Child Nodes of $P$}
        if All Nodes $i \in H^P$ are Leaf Nodes then      ▷ A Leaf Node is a Node with 0 Child Nodes
            $|H^P_i|$ = The sum of all Class Counts in $H^P_i$          ▷ Each Node in $H^P$ has a $N^C_D$
            $|P| = \sum_{i \in H_P} |H^P_i|$
            if $\sum_{i \in H^P} \frac{|H^P_i|}{|P|} \times G(H^P_i) < G(P)$ then   ▷ We only need $N^C_D$ to calculate the Gini Index
                Remove $H^P$ from $P$                                   ▷ $P$ is now a leaf node
                PruneTree($T$ with updated $P$)
            end if
        end if
    end for
    return $T$
end procedure
```

5 Experiments and Results

Using six datasets from the UCI Machine Learning Repository (Bache & Lichman 2013), we compare the prediction accuracy of our proposed algorithm, DPDF, to DiffPID3 (Friedman & Schuster 2010). We implement the main version of DiffPID3 (Friedman & Schuster 2010) (that uses the Exponential Mechanism), and avoid datasets with continuous attributes so that DiffID3 is not disadvantaged by its expensive usage of the privacy budget β for continuous attributes. As a benchmark to demonstrate the potential the chosen datasets have for classification, we provide the prediction accuracy results of the popular Random Forest algorithm developed by Breiman (2001). We build the Random Forest using all of the default parameter settings in sci-kit learn 0.15.2 (Pedregosa et al. 2011). All reported prediction accuracies for all algorithms are the average prediction accuracy results of performing 10 iterations of stratified 10-fold cross-validation. This involves randomly dividing each dataset into 10 equal partitions in a way that keeps the class distribution in each partition as close to the whole dataset as possible. Nine of the partitions are then combined to make D and are used to build the classifier (whether it is DPDF, DiffPID3 or Random Forest). The final partition is used as the testing data B. This is repeated with all 10 combinations of nine partitions, so that each partition is used as B once. Ten iterations of this cross-validation process are performed, with the partitions being randomly generated each time. This means that 100 prediction accuracy results are produced, and the average of these are what we report in all our figures.

We test two parameter settings for our DPDF: where $\tau = 1$ and where $\tau = 4$. For all experiments with DPDF or DiffPID3, $\delta = 5$. We use 5 values of β for our experiments: 0.1, 0.25, 0.5, 1.0 and 2.0. Note that no privacy preservation of any kind is applied to Random Forest.

The datasets used are all publicly available and have the following names in the UCI Machine Learning Repository: “Car Evaluation” (Figure 2), “Chess (King-Rook vs. King)” (Figure 3), “Connect4” (Figure 4), “Mushroom” (Figure 5), “Nursery” (Figure 6), and “Tic-Tac-Toe Endgame” (Figure 7). The number of records in each dataset ranges from 958 to 67557; the number of attributes ranges from six to 42; the number of class values ranges from two to 18.

For most datasets, our algorithm halves the difference between DiffPID3’s prediction accuracy and Random Forest’s prediction accuracy. The biggest improvement over DiffID3 is seen with the Nursery dataset, where DPDF with $\tau = 1$ beats DiffPID3 by almost 25 percentage points, and comes within 7 percent percentage points of Random Forest. With the Tic-Tac-Toe and Connect4 datasets, we can see some overlap between DiffPID3 and DPDF when $\tau = 1$, however in both cases, DPDF always beats DiffPID3 when $\tau = 4$, indicating the benefit of having multiple trees. Interestingly, sometimes $\tau = 1$ beats $\tau = 4$; this may be due to the lower ϵ value available to each query Q when $\tau = 4$, leading to more noisy outputs. In Table 1, we demonstrate the difference between the size of ϵ for DiffPID3 and DPDF when $\tau = 1$ and $\tau = 4$, for each of the five β values we are testing.

As expected, the prediction accuracy of DiffPID3 and DPDF (for both $\tau = 1$ and $\tau = 4$) generally increases as the budget β increases. In a few situations this does not happen, notably for the Connect4 dataset. However in this dataset, Random Forest produces a worse prediction accuracy than all the differentially private algorithms at all β values, suggesting that the noisy outputs of DiffPID3 and DPDF helped the tree-building process. This may be due to the trees in Random Forest getting stuck in local optima, or simply that decision trees are not a good choice for data mining the Connect4 dataset. Prediction Accuracy behaving in this way when applying privacy-preservation techniques has been explored by previous work (Fletcher & Islam 2014).

Overall, it appears that for most datasets (except Tic-Tac-Toe, perhaps because it is by far the smallest dataset and thus has the noisiest outputs), DPDF produces a decision forest of acceptable quality for most data mining needs. This is true even when β is very low; as low as $\beta = 0.1$, where each query Q has $\epsilon = 0.003$ when $\tau = 4$. The user also has the complete list of rules and sub-rules from F, including the class distribution of each rule. This means the user can easily remove any rules or sub-rules with low confidence, leaving them with a shorter list of high quality rules.

Technique	**Privacy Budget β**				
	0.1	**0.25**	**0.5**	**1.0**	**2.0**
DiffPID3	0.010	0.025	0.050	0.100	0.200
DPDF, $\tau = 1$	0.011	0.028	0.056	0.111	0.222
DPDF, $\tau = 4$	0.003	0.007	0.014	0.028	0.056

Table 1: The size of ϵ per query, depending on the technique used and the total privacy budget β. In the example shown, the depth of the trees is $\delta = 5$.

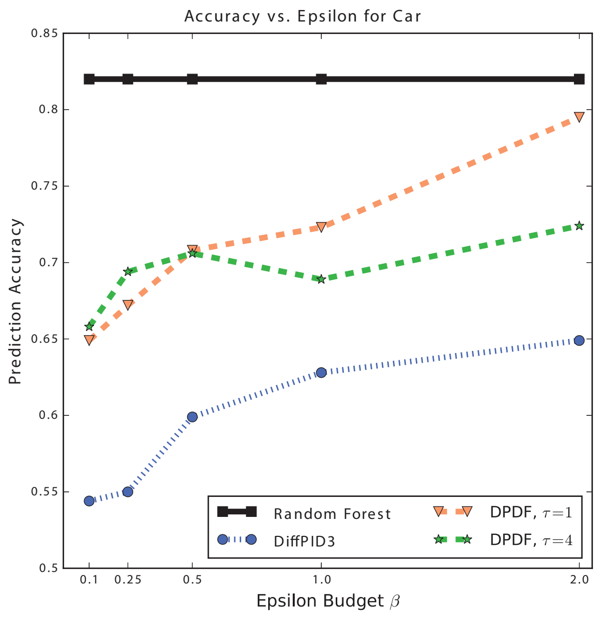

Figure 2: A comparison of our technique (DPDF) to DiffPID3 using the Car dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

6 Conclusion

The success of DPDF at even very low β values provides the user with some valuable options – instead of using their entire allocated privacy budget β on a single run of DPDF, they can instead use only a fraction of it. The rest of the budget could be spent on anything the user wishes. This might include some preliminary queries on D in order to tune the δ and τ parameters, which would be an interesting direction

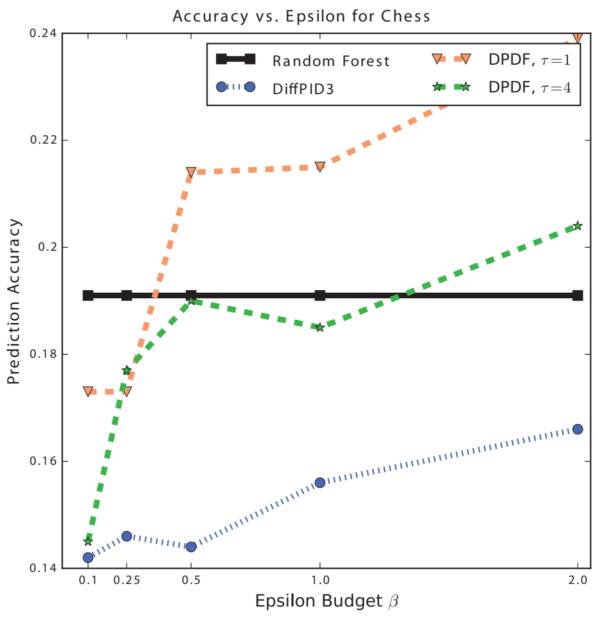

Figure 3: A comparison of our technique (DPDF) to DiffPID3 using the Chess dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$. Note that there are 18 class values, so randomly guessing would give a prediction accuracy of 5.55%; hence the low Prediction Accuracy.

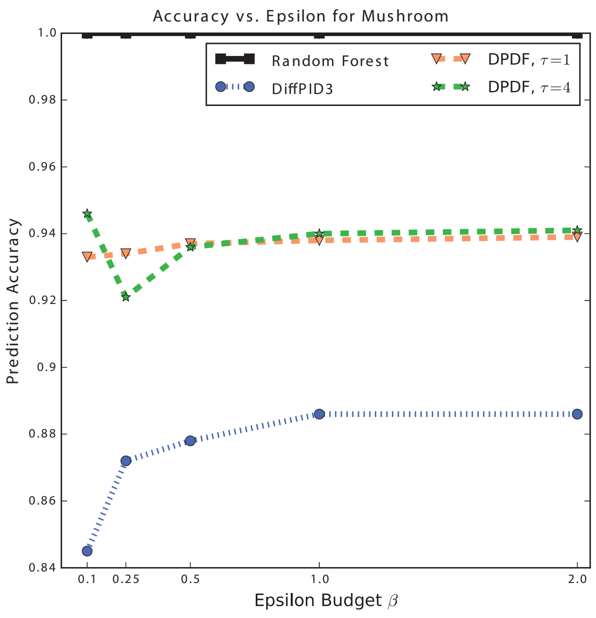

Figure 5: A comparison of our technique (DPDF) to DiffPID3 using the Mushroom dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

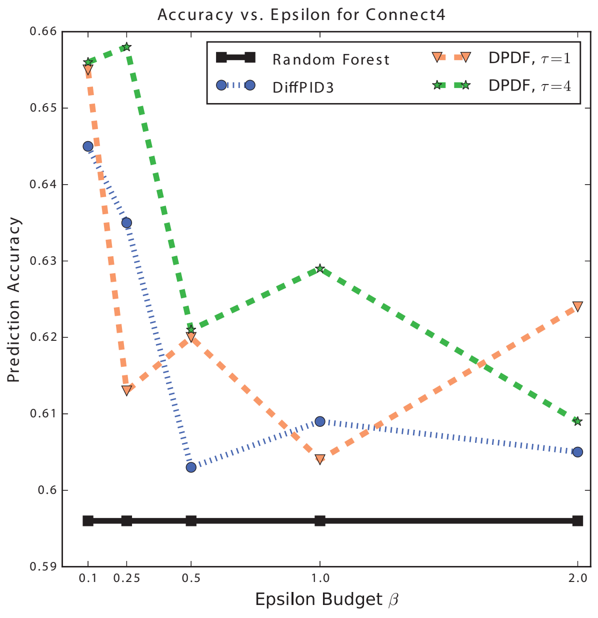

Figure 4: A comparison of our technique (DPDF) to DiffPID3 using the Connect4 dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

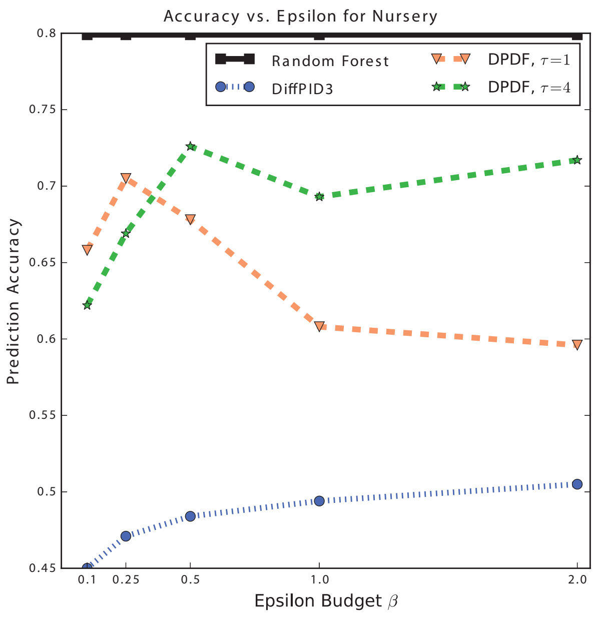

Figure 6: A comparison of our technique (DPDF) to DiffPID3 using the Nursery dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

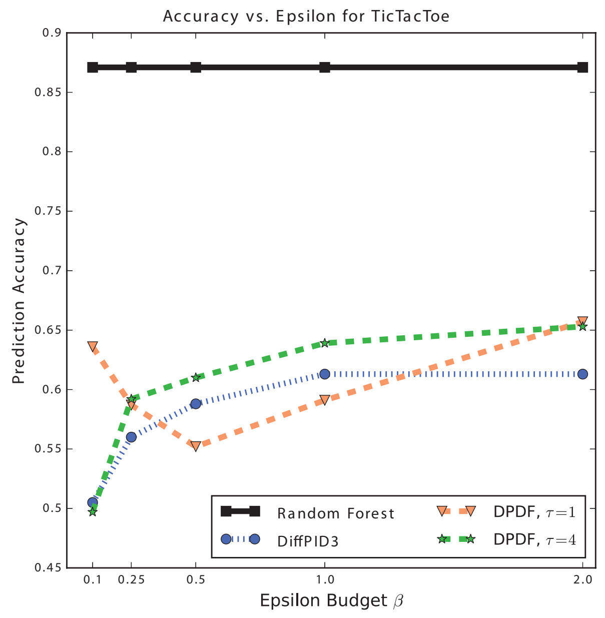

Figure 7: A comparison of our technique (DPDF) to DiffPID3 using the Tic-Tac-Toe dataset, with Random Forest included as a benchmark. We test two parameter settings for DPDF: $\tau = 1$ and $\tau = 4$.

for future research. It could include multiple runs of DPDF, each with different parameters. It could include completely different data mining algorithms, such as clustering (as long as it is differentially private). The strong mathematical properties of differential privacy allow the data owner to guarantee the individuals in the dataset that their presence in the dataset is almost completely undetectable, no matter how a user decides to divide their β budget. Our novel theorem on the local sensitivity of the Gini Index will be useful to any future classification algorithms that wish to perform differentially private data mining. We provide the code required to implement DPDF, so that data miners and fellow researchers may take advantage of it.

References

Bache, K. & Lichman, M. (2013), 'UCI Machine Learning Repository'.
URL: *http://archive.ics.uci.edu/ml/*

Breiman, L. (2001), 'Random forests', *Machine learning* pp. 1–35.

Breiman, L., Friedman, J., Stone, C. & Olshen, R. (1984), *Classification and regression trees*, Chapman & Hall/CRC.

Dwork, C. (2006), Differential Privacy, *in* 'Automata, languages and programming', Vol. 4052, Springer Berlin Heidelberg, Venice, Italy, pp. 1–12.

Dwork, C. (2007), 'An Ad Omnia Approach to Defining and Achieving Private Data Analysis'.

Dwork, C. (2008), 'Differential Privacy: A survey of results', *Theory and Applications of Models of Computation* pp. 1–19.

Dwork, C. (2011), 'A firm foundation for private data analysis', *Communications of the ACM* **54**(1), 86–95.

Dwork, C., McSherry, F., Nissim, K. & Smith, A. (2006), 'Calibrating noise to sensitivity in private data analysis', *Theory of Cryptography* pp. 265–284.

Dwork, C. & Roth, A. (2014), *The Algorithmic Foundations of Differential Privacy*, Vol. 9, Now Publishers.

Fletcher, S. & Islam, M. Z. (2014), Quality evaluation of an anonymized dataset, *in* '22nd International Conference on Pattern Recognition', IEEE, Stockholm, Sweden, pp. 3594–3599.

Fletcher, S. & Islam, M. Z. (2015), 'An Anonymization Technique using Intersected Decision Trees', *Journal of King Saud University - Computer and Information Sciences* **27**(3).

Friedman, A. & Schuster, A. (2010), Data Mining with Differential Privacy, *in* '16th SIGKDD Conference on Knowledge Discovery and Data Mining', ACM, Washington, DC, USA, pp. 493–502.

Fung, B., Wang, K., Chen, R. & Yu, P. (2010), 'Privacy-preserving data publishing: A survey of recent developments', *ACM Computing Surveys (CSUR)* **42**(4), 14.

Islam, M. Z. & Giggins, H. (2011), Knowledge discovery through SysFor: a systematically developed forest of multiple decision trees, *in* 'Ninth Australasian Data Mining Conference-Volume 121', Australian Computer Society, Inc., Ballarat, Australia, pp. 195–204.

Kotsiantis, S. & Kanellopoulos, D. (2006), 'Discretization techniques: A recent survey', *GESTS International Transactions on Computer Science and Engineering* **32**(1), 47–58.

LeFevre, K., DeWitt, D. & Ramakrishnan, R. (2005), Incognito: Efficient full-domain k-anonymity, *in* 'Proceedings of the 2005 ACM SIGMOD international conference on Management of data', ACM, pp. 49–60.

McSherry, F. (2009), Privacy integrated queries: an extensible platform for privacy-preserving data analysis, *in* 'Proceedings of the 35th SIGMOD international conference on Management of data', ACM, Providence, USA, pp. 19–30.

McSherry, F. & Talwar, K. (2007), 'Mechanism Design via Differential Privacy', *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)* pp. 94–103.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Quinlan, J. R. (1993), *C4.5: programs for machine learning*, 1st edn, Morgan kaufmann.

Sweeney, L. (2002), 'Achieving k-anonymity privacy protection using generalization and suppression', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **10**(5), 571–588.

UN General Assembly (1948), 'Universal Declaration of Human Rights'.