Data Mining with Differential Privacy

Arik Friedman and Assaf Schuster
Technion - Israel Institute of Technology
Haifa 32000, Israel
{arikf,assaf}@cs.technion.ac.il

ABSTRACT

We consider the problem of data mining with formal privacy guarantees, given a data access interface based on the differential privacy framework. Differential privacy requires that computations be insensitive to changes in any particular individual's record, thereby restricting data leaks through the results. The privacy preserving interface ensures unconditionally safe access to the data and does not require from the data miner any expertise in privacy. However, as we show in the paper, a naive utilization of the interface to construct privacy preserving data mining algorithms could lead to inferior data mining results. We address this problem by considering the privacy and the algorithmic requirements simultaneously, focusing on decision tree induction as a sample application. The privacy mechanism has a profound effect on the performance of the methods chosen by the data miner. We demonstrate that this choice could make the difference between an accurate classifier and a completely useless one. Moreover, an improved algorithm can achieve the same level of accuracy and privacy as the naive implementation but with an order of magnitude fewer learning samples.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.2.7 [**Database Management**]: Database Administration—*Security, Integrity and Protection*

General Terms

Algorithms, Security

Keywords

Differential Privacy, Data Mining, Decision Trees

1. INTRODUCTION

Data mining presents many opportunities for enhanced services and products in diverse areas such as healthcare, banking, traffic planning, online search, and so on. However, its promise is hindered by concerns regarding the privacy of the individuals whose data are being mined. Therefore, there is great value in data mining solutions that provide reliable privacy guarantees without significantly compromising accuracy. In this work we consider data mining within the framework of differential privacy [7, 9]. Basically, differential privacy requires that computations be insensitive to changes in any particular individual's record. Once an individual is certain that his or her data will remain private, being opted in or out of the database should make little difference. For the data miner, however, all these individual records in aggregate are very valuable. Differential privacy provides formal privacy guarantees that do not depend on an adversary's background knowledge or computational power. This independence frees data providers who share data from concerns about past or future data releases and is adequate given the abundance of personal information shared on social networks and public Web sites. In addition, differential privacy maintains composability [12] – differential privacy guarantees can be provided even when multiple differentially-private releases are available to an adversary. Thus, data providers that let multiple parties access their database can evaluate and limit any privacy risks that might arise from collusion between adversarial parties or due to repetitive access by the same party.

We consider the enforcement of differential privacy through a programmable privacy preserving layer, similar to the Privacy INtegrated Queries platform (PINQ) proposed by McSherry [15]. This concept is illustrated in Figure 1. In this approach, a data miner can access a database through a query interface exposed by the privacy preserving layer. The data miner need not worry about enforcing the privacy requirements nor be an expert in the privacy domain. The access layer enforces differential privacy by adding carefully calibrated noise to each query. Depending on the calculated function, the magnitude of noise is chosen to mask the influence of any particular record on the outcome. This approach has two useful advantages in the context of data mining: it allows data providers to outsource data mining tasks without exposing the raw data, and it allows data providers to sell data access to third parties while limiting privacy risks.

However, while the data access layer ensures that privacy is maintained, the implementation choices made by the data miner are crucial to the accuracy of the resulting data min-

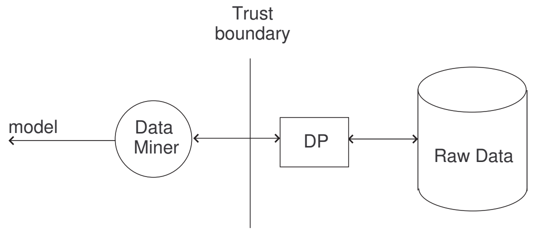

Figure 1: Data mining with differential privacy (DP) access interface

ing model. In fact, a straightforward adaptation of data mining algorithms to work with the privacy preserving layer could lead to suboptimal performance. Each query introduces noise to the calculation and different functions may require different magnitudes of noise to maintain the differential privacy requirements set by the data provider. Poor implementation choices could introduce larger magnitudes of noise than necessary, leading to inaccurate results.

To illustrate this problem further, we present it in terms of Pareto efficiency [21]. Consider three objective functions: the accuracy of the data mining model (e.g., the expected accuracy of a resulting classifier, estimated by its performance on test samples), the size of the mined database (number of training samples), and the privacy requirement, represented by a privacy parameter ϵ. In a given situation, one or more of these factors may be fixed: a client may present a lower acceptance bound for the accuracy of a classifier, the database may contain a limited number of samples, or a regulator may pose privacy restrictions. Within the given constraints, we wish to improve the objective functions: achieve better accuracy with fewer learning examples and better privacy guarantees. However, these objective functions are often in conflict. For example, applying stronger privacy guarantees could reduce accuracy or require a larger dataset to maintain the same level of accuracy. Instead, we should settle for some tradeoff. With this perception in mind, we can evaluate the performance of data mining algorithms. Consider, for example, three hypothetical algorithms that produce a classifier. Assume that their performance was evaluated on datasets with 50,000 records, with the results illustrated in Figure 2. We can see that when the privacy settings are high, algorithm 1 obtains on average a lower error rate than the other algorithms, while algorithm 2 does better when the privacy settings are low. A *Pareto improvement* is a change that improves one of the objective functions without harming the others. Algorithm 3 is dominated by the other algorithms: for any setting, we can make a Pareto improvement by switching to one of the other algorithms. A given situation (a point in the graph) is *Pareto efficient* when no further Pareto improvements can be made. The *Pareto frontier* is given by all the Pareto efficient points. Our goal is to investigate algorithms that can further extend the Pareto frontier, allowing for better privacy and accuracy tradeoffs.

We address this problem by considering the privacy and algorithmic requirements simultaneously, taking decision tree induction as a case study. We consider how algorithmic processes such as the application of splitting criteria on nominal and continuous attributes as well as decision tree pruning, and investigate how privacy considerations may influence the way the data miner utilizes the data access interface. Similar considerations motivate the extension of the interface with additional functionality that allows the data miner to query the database more effectively. Our analysis and experimental evaluations confirm that algorithmic decisions made with privacy considerations in mind may have a profound impact on the resulting classifier. When differential privacy is applied, the method chosen by the data miner to build the classifier could make the difference between an accurate classifier and a useless one, even when the same choice without privacy constraints would have no such effect. Moreover, an improved algorithm can achieve the same level of accuracy and privacy as the naive implementation but with an order of magnitude fewer learning samples.

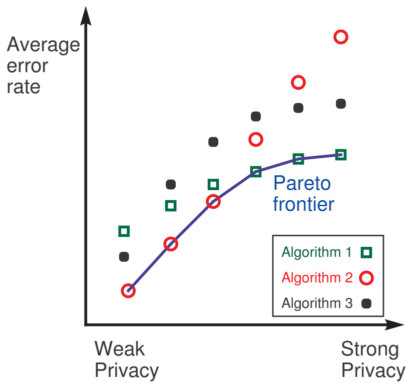

Figure 2: Example of a Pareto frontier. Given a number of learning samples, what are the privacy and accuracy tradeoffs?

1.1 Related Work

Most of the research on differential privacy so far focused on theoretical properties of the model, providing feasibility and infeasibility results [13, 10, 2, 8].

Several recent works studied the use of differential privacy in practical applications. Machanavajjhala et al. [14] applied a variant of differential privacy to create synthetic datasets from U.S. Census Bureau data, with the goal of using them for statistical analysis of commuting patterns in mapping applications. Chaudhuri and Monteleoni [4] proposed differentially-private algorithms for logistic regression. These algorithms ensure differential privacy by adding noise to the outcome of the logistic regression model or by solving logistic regression for a noisy version of the target function. Unlike the approach considered in this paper, the algorithms require direct access to the raw data. McSherry and Mironov studied the application of differential privacy to collaborative recommendation systems [16] and demonstrated the feasibility of differential privacy guarantees without a significant loss in recommendation accuracy.

The use of synthetic datasets for privacy preserving data analysis can be very appealing for data mining applications, since the data miner gets unfettered access to the synthetic dataset. This approach was studied in several works [2, 11, 10]. Unfortunately, initial results suggest that ensuring the usefulness of the synthetic dataset requires that it be crafted to suit the particular type of analysis to be performed.

2. BACKGROUND

2.1 Differential Privacy

Differential privacy [7, 8] is a recent privacy definition that guarantees the outcome of a calculation to be insensitive to any particular record in the data set.

Definition 1. We say a randomized computation M provides ϵ-differential privacy if for any datasets A and B with symmetric difference $A \Delta B = 1$ (A and B are treated as multisets), and any set of possible outcomes $S \subseteq Range(M)$,

$$\Pr[M(A) \in S] \leq \Pr[M(B) \in S] \times e^{\epsilon} \ .$$

The parameter ϵ allows us to control the level of privacy. Lower values of ϵ mean stronger privacy, as they limit further the influence of a record on the outcome of a calculation. The values typically considered for ϵ are smaller than 1 [8], e.g., 0.01 or 0.1 (for small values we have $e^{\epsilon} \approx 1 + \epsilon$). The definition of differential privacy maintains a *composability* property [15]: when consecutive queries are executed and each maintains differential privacy, their ϵ parameters can be accumulated to provide a differential privacy bound over all the queries. Therefore, the ϵ parameter can be treated as a privacy cost incurred when executing the query. These costs add up as more queries are executed, until they reach an allotted bound set by the data provider (referred to as the privacy budget), at which point further access to the database will be blocked. The composition property also provides some protection from collusion: collusion between adversaries will not lead to a direct breach in privacy, but rather cause it to degrade gracefully as more adversaries collude, and the data provider can also bound the overall privacy budget (over all data consumers).

Typically, differential privacy is achieved by adding noise to the outcome of a query. One way to do so is by calibrating the magnitude of noise required to obtain ϵ-differential privacy according to the *sensitivity* of a function [9]. The sensitivity of a real-valued function expresses the maximal possible change in its value due to the addition or removal of a single record:

Definition 2. Given a function $f : D \rightarrow \mathbb{R}^d$ over an arbitrary domain D, the sensitivity of f is

$$S(f) = \max_{A,B \text{ where } A \Delta B = 1} \|f(A) - f(B)\|_1 \ .$$

Given the sensitivity of a function f, the addition of noise drawn from a calibrated Laplace distribution maintains ϵ-differential privacy [9]:

THEOREM 1. *Given a function $f : D \rightarrow \mathbb{R}^d$ over an arbitrary domain D, the computation*

$$M(X) = f(X) + (Laplace(S(f)/\epsilon))^d$$

provides ϵ-differential privacy.

For example, the count function over a set S, $f(S) = |S|$, has sensitivity 1. Therefore, a noisy count that returns $M(S) = |S| + Laplace(1/\epsilon)$ maintains ϵ-differential privacy. Note that the added noise depends in this case only on the privacy parameter ϵ. Therefore, the larger the set S, the smaller the relative error introduced by the noise.

Another way to obtain differential privacy is through the exponential mechanism [17]. The exponential mechanism is given a quality function q that scores outcomes of a calculation, where higher scores are better. For a given database and ϵ parameter, the quality function induces a probability distribution over the output domain, from which the exponential mechanism samples the outcome. This probability distribution favors high scoring outcomes (they are exponentially more likely to be chosen), while ensuring ϵ-differential privacy.

Definition 3. Let $q : (\mathcal{D}^n \times \mathcal{R}) \rightarrow \mathbb{R}$ be a quality function that, given a database $d \in \mathcal{D}^n$, assigns a score to each outcome $r \in \mathcal{R}$. Let $S(q) = \max_{r, A \Delta B = 1} \|q(A,r) - q(B,r)\|_1$. Let M be a mechanism for choosing an outcome $r \in \mathcal{R}$ given a database instance $d \in D^n$. Then the mechanism M, defined by

$$M(d,q) = \left\{ \text{return } r \text{ with probability} \propto \exp\left(\frac{\epsilon q(d,r)}{2S(q)}\right) \right\} \ ,$$

maintains ϵ-differential privacy.

2.2 PINQ

PINQ [15] is a proposed architecture for data analysis with differential privacy. It presents a wrapper to C#'s LINQ language for database access, and this wrapper enforces differential privacy. A data provider can allocate a privacy budget (parameter ϵ) for each user of the interface. The data miner can use this interface to execute over the database aggregate queries such as count (`NoisyCount`), sum (`NoisySum`) and average (`NoisyAvg`), and the wrapper uses Laplace noise and the exponential mechanism to enforce differential privacy.

Another operator presented in PINQ is `Partition`. When queries are executed on disjoint datasets, the privacy costs do not add up, because each query pertains to a different set of records. This property was dubbed *parallel composition* [15]. The `Partition` operator takes advantage of parallel composition: it divides the dataset into multiple disjoint sets according to a user-defined function, thereby signaling to the system that the privacy costs for queries performed on the disjoint sets should be summed separately. Consequently, the data miner can utilize the privacy budget more efficiently.

Note that a data miner wishing to develop a data mining algorithm using the privacy preserving interface should plan ahead the number of queries to be executed and the value of ϵ to request for each. Careless assignment of privacy costs to queries could lead to premature exhaustion of the privacy budget set by the data provider, thereby blocking access to the database half-way through the data mining process.

3. A SIMPLE PRIVACY PRESERVING ID3

The input to the decision tree induction algorithm is a dataset $\mathcal{T}$ with attributes $\mathcal{A} = \{A_1, \ldots, A_d\}$ and a class attribute C. Each record in the dataset pertains to one individual. The goal is to build a classifier for the class C. The ID3 algorithm presented by Quinlan [19] uses greedy hill-climbing to induce a decision tree. Initially, the root holds all the learning samples. Then, the algorithm chooses the attribute that maximizes the information gain and splits the learning samples with this attribute. The same process is applied recursively on each subset of the learning samples, until there are no further splits that improve the information gain. Algorithm 1 presents a differential privacy adaptation of ID3, which evaluates the information gain using noisy counts (i.e., adding Laplace noise to the accurate count). It is based on a theoretical algorithm that was presented in the SuLQ framework (Sub-Linear Queries [1]), a predecessor of differential privacy, so we will refer to it as SuLQ-based ID3.

We use the following notation: $\mathcal{T}$ refers to a set of records, $\tau = |\mathcal{T}|$, r_A and r_C refer to the values that record $r \in \mathcal{T}$ takes on the attributes A and C respectively, $\mathcal{T}_j^A = \{r \in \mathcal{T} : r_A = j\}$, $\tau_j^A = |\mathcal{T}_j^A|$, $\tau_c = |r \in \mathcal{T} : r_C = c|$, and $\tau_{j,c}^A = |r \in \mathcal{T} : r_A = j \wedge r_C = c|$. To refer to noisy counts, we use a similar notation but substitute N for τ. All the log() expressions are in base 2.

Algorithm 1 SuLQ-based ID3

```
1: procedure SuLQ_ID3($\mathcal{T}, \mathcal{A}, C, d, B$)
2:     Input: $\mathcal{T}$ – private dataset, $\mathcal{A} = \{A_1, \ldots, A_d\}$ – a set of attributes, $C$ – class attribute, $d$ – maximal tree depth, $B$ – differential privacy budget
3:     $\epsilon = \frac{B}{2(d+1)}$
4:     Build_SuLQ_ID3($\mathcal{T}, \mathcal{A}, C, d, \epsilon$)
5: end procedure
6: procedure Build_SuLQ_ID3($\mathcal{T}, \mathcal{A}, C, d, \epsilon$)
7:     $t = \max_{A \in \mathcal{A}} |A|$
8:     $N_{\mathcal{T}} = \texttt{NoisyCount}_\epsilon(\mathcal{T})$
9:     if $\mathcal{A} = \emptyset$ or $d = 0$ or $\frac{N_{\mathcal{T}}}{t|C|} < \frac{\sqrt{2}}{\epsilon}$ then
10:        $\mathcal{T}_c = \texttt{Partition}(\mathcal{T}, \forall c \in C : r_C = c)$
11:        $\forall c \in C : N_c = \texttt{NoisyCount}_\epsilon(\mathcal{T}_c)$
12:        return a leaf labeled with $\arg\max_c(N_c)$
13:    end if
14:    for every attribute $A \in \mathcal{A}$ do
15:        $\mathcal{T}_j = \texttt{Partition}(\mathcal{T}, \forall j \in A : r_A = j)$
16:        $\forall j \in A : \mathcal{T}_{j,c} = \texttt{Partition}(\mathcal{T}_j, \forall c \in C : r_C = c)$
17:        $N_j^A = \texttt{NoisyCount}_{\epsilon/(2|\mathcal{A}|)}(\mathcal{T}_j)$
18:        $N_{j,c}^A = \texttt{NoisyCount}_{\epsilon/(2|\mathcal{A}|)}(\mathcal{T}_{j,c})$
19:        $\overline{V_A} = \sum_{j=1}^{|A|} \sum_{c=1}^{|C|} N_{j,c}^A \cdot \log \frac{N_{j,c}^A}{N_j^A}$ (negative $N_j^A$ or $N_{j,c}^A$ are skipped)
20:    end for
21:    $\bar{A} = \arg\max_A \overline{V_A}$
22:    $\mathcal{T}_i = \texttt{Partition}(\mathcal{T}, \forall i \in \bar{A} : r_{\bar{A}} = i)$
23:    $\forall i \in \bar{A}$ : Subtree$_i$ = $Build\_SuLQ\_ID3(\mathcal{T}_i, \mathcal{A} \setminus \bar{A}, C, d-1, \epsilon)$.
24:    return a tree with a root node labeled $\bar{A}$ and edges labeled 1 to $|\bar{A}|$ each going to Subtree$_i$
25: end procedure
```

Given the entropy of a set of instances $\mathcal{T}$ with respect to the class attribute C, $H_C(\mathcal{T}) = -\sum_{c \in C} \frac{\tau_c}{\tau} \log \frac{\tau_c}{\tau}$ and given the entropy obtained by splitting the instances with attribute A, $H_{C|A}(\mathcal{T}) = \sum_{j \in A} \frac{\tau_j^A}{\tau} \cdot H_C(\mathcal{T}_j^A)$, the information gain is given by InfoGain$(A, \mathcal{T}) = H_C(\mathcal{T}) - H_{C|A}(\mathcal{T})$. Maximizing the information gain is equivalent to maximizing

$$V(A) = -\tau \cdot H_{C|A}(\mathcal{T}) = \sum_{j \in A} \tau_j^A \cdot H_C(\mathcal{T}_j^A) \ . \quad (1)$$

Hence, information gain can be approximated with noisy counts for τ_j^A and $\tau_{j,c}^A$ to obtain:

$$\overline{V_A} = \sum_{j=1}^{|A|} \sum_{c=1}^{|C|} N_{j,c}^A \cdot \log \frac{N_{j,c}^A}{N_j^A} \ . \quad (2)$$

In ID3, when all the instances in a node have the same class or when no instances have reached a node, there will be no further splits. Because of the noise introduced by differential privacy, these stopping criteria can no longer be evaluated reliably. Instead, in line 9 we try to evaluate whether there are "enough" instances in the node to warrant further splits. While the theoretical version of SuLQ-based ID3 in [1] provided bounds with approximation guarantees, we found these bounds to be prohibitively large. Instead, our heuristic requirement is that, in the subtrees created after a split, each class count be larger on average than the standard deviation of the noise in the `NoisyCount`. While this requirement is quite arbitrary, in the experiments it provided reasonable results.

Use of the overall budget B set by the data provider should be planned ahead. The SuLQ-based ID3 algorithm does that by limiting the depth of the tree according to a value set by the data miner and assigning an equal share of the budget for each level of the tree, including the leaves. Thanks to the composition property of differential privacy, queries on different nodes on the same level do not accumulate, as they are carried out on disjoint sets of records. Within each node, half of the allocated budget is used to evaluate the number of instances and the other half is used to determine the class counts (in leaves) or evaluate the attributes (in nodes). Class counts are calculated on disjoint sets, so each query can use the allocated ϵ. On the other hand, because each attribute evaluation is carried out on the same set of records, the budget must be further split among the attributes.

The suggested implementation in SuLQ-based ID3 is relatively straightforward: it makes direct use of the `NoisyCount` primitive to evaluate the information gain criterion, while taking advantage of the `Partition` operator to avoid redundant accumulation of the privacy budget. However, this implementation also demonstrates the drawback of a straightforward adaptation of the algorithm: because the count estimates required to evaluate the information gain should be carried out for each attribute separately, the data miner needs to split the overall budget between those separate queries. Consequently, the budget per query is small, resulting in large magnitudes of noise which must be compensated for by larger datasets.

4. PRIVACY CONSIDERATIONS IN DECISION TREE INDUCTION

4.1 Splitting Criteria

The main drawback of the SuLQ-based ID3 presented in the previous section is the wasteful use of the privacy budget when the information gain should be evaluated separately for each attribute. The exponential mechanism offers a better approach: rather than evaluating each attribute separately, we can evaluate the attributes simultaneously in one query, the outcome of which is the attribute to use for splitting. The quality function q provided to the exponential mechanism scores each attribute according to the splitting criterion. Algorithm 2 presents this approach. The budget distribution in the algorithm is similar to that used for the SuLQ-based ID3, except that for attribute selection, instead of splitting the allocated budget ϵ among multiple queries, the entire budget is used to find the best attribute in a single exponential mechanism query.

Algorithm 2 Differential Private ID3 algorithm

```
1: procedure DiffPID3($\mathcal{T}, \mathcal{A}, C, d, B$)
2:     Input: $\mathcal{T}$ – private dataset, $\mathcal{A} = \{A_1, \ldots, A_d\}$ – a set of attributes, $C$ – class attribute, $d$ – maximal tree depth, $B$ – differential privacy budget
3:     $\epsilon = \frac{B}{2(d+1)}$
4:     Build_DiffPID3($\mathcal{T}, \mathcal{A}, C, d, \epsilon$)
5: end procedure
6: procedure Build_DiffPID3($\mathcal{T}, \mathcal{A}, C, d, \epsilon$)
7:     $t = \max_{A \in \mathcal{A}} |A|$
8:     $N_{\mathcal{T}} = \texttt{NoisyCount}_\epsilon(T)$
9:     if $\mathcal{A} = \emptyset$ or $d = 0$ or $\frac{N_{\mathcal{T}}}{t|C|} < \frac{\sqrt{2}}{\epsilon}$ then
10:        $\mathcal{T}_c = \texttt{Partition}(\mathcal{T}, \forall c \in [C] : r_C = c)$
11:        $\forall c \in C : N_c = \texttt{NoisyCount}_\epsilon(\mathcal{T}_c)$
12:        return a leaf labeled with $\arg\max_c(N_c)$
13:    end if
14:    $\bar{A} = \texttt{ExpMech}_\epsilon(\mathcal{A}, q)$
15:    $\mathcal{T}_i = \texttt{Partition}(\mathcal{T}, \forall i \in \bar{A} : r_{\bar{A}} = i)$
16:    $\forall i \in \bar{A}$ : Subtree$_i$ = $Build\_DiffPID3(\mathcal{T}_i, A \setminus \bar{A}, C, d-1, \epsilon)$.
17:    return a tree with a root node labeled $\bar{A}$ and edges labeled 1 to $|\bar{A}|$ each going to Subtree$_i$.
18: end procedure
```

The next question is which quality function should be fed into the exponential mechanism. Although many studies have compared the performance of different splitting criteria for decision tree induction, their results do not, in general, testify to the superiority of any one criterion in terms of tree accuracy, although the choice may affect the resulting tree size (see, e.g., [18]). Things change, however, when the splitting criteria are considered in the context of algorithm 2. First, the depth constraint may prevent some splitting criteria from inducing trees with the best possible accuracy. Second, the sensitivity of the splitting criterion influences the magnitude of noise introduced to the exponential mechanism, meaning that for the same privacy parameter, the exponential mechanism will have different effectiveness for different splitting criteria. We consider several quality functions and their sensitivity. The proofs for the sensitivity bounds are supplied in Appendix A:

Information gain: following the discussion leading to equation 1, we take the quality function for information gain to be

$$q_{\text{IG}}(\mathcal{T}, A) = V(A) = -\sum_{j \in A} \sum_{c \in C} \tau_{j,c}^A \cdot \log \frac{\tau_{j,c}^A}{\tau_j^A} \ .$$

The sensitivity of this function is $S(q_{\text{IG}}) = \log(N + 1) + 1/\ln 2$, where N is a bound on the dataset size. We assume that such a bound is known or given by the data provider.

Gini index: this impurity measure is used in the CART algorithm [3]. It denotes the probability to incorrectly label a sample when the label is picked randomly according to the distribution of class values for an attribute value t. Minimizing the Gini index is equivalent to maximizing the following quality function:

$$q_{\text{Gini}}(\mathcal{T}, A) = -\sum_{j \in A} \tau_j^A \left(1 - \sum_{c \in C} \left(\frac{\tau_{j,c}^A}{\tau_j^A} \right)^2 \right) \ .$$

The sensitivity of this function is $S(q_{\text{Gini}}) = 2$.

Max operator: (based on the resubstitution estimate described in [3]) this function corresponds to the node misclassification rate by picking the class with the highest frequency:

$$q_{\text{Max}}(\mathcal{T}, A) = \sum_{j \in A} \left(\max_c (\tau_{j,c}^A) \right) \ .$$

The sensitivity of this function is $S(q_{\text{Max}}) = 1$.

Gain Ratio: the gain ratio [20] is obtained by dividing the information gain by a measure called *information value*, defined as $IV(A) = -\sum_{j \in A} \frac{\tau_j^A}{\tau} \cdot \log \frac{\tau_j^A}{\tau}$. Unfortunately, when $IV(A)$ is close to zero (happens when $\tau_j^A \approx \tau$), the gain ratio may become undefined or very large. This known problem is circumvented in C4.5 by calculating the gain ratio only for a subset of attributes that are above the average gain. The implication is that the sensitivity of the gain ratio cannot be bounded and consequently, the gain ratio cannot be usefully applied with the exponential mechanism.

The sensitivity of the quality functions listed above suggests that information gain will be the most sensitive to noise, and the Max operator will be the least sensitive to noise. In the experimental evaluations we compare the performance of these quality functions, and the influence of the noise is indeed reflected in the accuracy of the resulting classifiers.

4.2 Pruning

One problem that may arise when building classifiers is overfitting the training data. When inducing decision trees with differential privacy, this problem is somewhat mitigated by the introduction of noise and by the constraint on tree depth. Nonetheless, because of the added noise, it is no longer possible to identify a leaf with pure class values, so the algorithm will keep splitting nodes as long as there are enough instances and as long as the depth constraint is not reached. Hence, the resulting tree may contain redundant splits, and pruning may improve the tree.

We avoid pruning approaches that require the use of a validation set, such as the minimal cost complexity pruning applied by CART [3] or reduced error pruning [20], because they lead to a smaller training set, which in turn would be more susceptible to the noise introduced by differential privacy. Instead, we consider error based pruning [20], which is used in C4.5. In this approach, the training set itself is used to evaluate the performance of the decision tree before and after pruning. Since this evaluation is biased in favor of the training set, the method makes a pessimistic estimate for the test set error rate: it assumes that the error rate has binomial distribution, and it uses a certainty factor CF (by default taken to be 0.25) as a confidence limit to estimate a bound on the error rate from the error observed on the training set. C4.5 estimates the error rate in a given subtree (according to the errors in the leaves), in its largest branch (pruning by subtree raising), and the expected error if the subtree is turned into a leaf. The subtree is then replaced with the option that minimizes the estimated error.

Since error based pruning relies on class counts of instances, it should be straightforward to evaluate the error rates using noisy counts. The error in the subtree can be evaluated using the class counts in the leaves, which were obtained in the tree construction phase. To evaluate the error if a subtree is turned into a leaf, the counts in the leaves can be aggregated in a bottom-up manner to provide the counts in upper level nodes. However, this aggregation would also add up all the noise introduced in the leaves (i.e., leading to larger noise variance). Moreover, subtrees split with multi-valued attributes would aggregate much more noise than those with small splits, skewing the results. Executing new `NoisyCount` queries to obtain class counts in upper level nodes could provide more accurate results. However, this would require an additional privacy budget at the expense of the tree construction phase. For similar reasons, error estimations for subtree raising would also incur a toll on the privacy budget.

As a compromise, we avoid making additional queries on the dataset and instead use the information gathered during the tree construction to mitigate the impact of the noise. We make two passes over the tree: an initial top-down pass calibrates the total instance count in each level of the tree to match the count in the parent level. Then a second bottom-up pass aggregates the class counts and calibrates them to match the total instance counts from the first pass. Finally, we use the updated class counts and instance counts to evaluate the error rates just as in C4.5 and prune the tree. Algorithm 3 summarizes this approach. In the algorithm, $N_{\mathcal{T}}$ refers to the noisy instance counts that were calculated in algorithm 2 for each node T, and $N_{\rfloor}$ refers to the class counts for each class $c \in C$.

Algorithm 3 Pruning with Noisy Counts

```
 1: Input: $UT$ - an unpruned decision tree, $CF$ - Certainty
    factor
 2: procedure PRUNE($UT$)
 3:     TopDownCorrect($UT$, $UT.N_{\mathcal{T}}$)
 4:     BottomUpAggregate($UT$)
 5:     C4.5Prune($UT$,$CF$)
 6: end procedure

 7: procedure TOPDOWNCORRECT($T$,$fixedN_{\mathcal{T}}$)
 8:     $T.N_{\mathcal{T}} \leftarrow fixedN_{\mathcal{T}}$
 9:     if $T$ is not a leaf then
10:         $T_i \leftarrow \text{subtree}_i(T)$
11:         for all $T_i$ do
12:             $fixedN_{\mathcal{T}_i} \leftarrow T.N_{\mathcal{T}} \cdot \frac{T_i.N_{\mathcal{T}}}{\sum_i T_i.N_{\mathcal{T}}}$
13:             TopDownCorrect($T_i$,$fixedN_{\mathcal{T}_i}$)
14:         end for
15:     end if
16: end procedure

17: procedure BOTTOMUPAGGREGATE($T$)
18:     if $T$ is a leaf then
19:         for all $c \in C$ do
20:             $T.N_c \leftarrow T.N_{\mathcal{T}} \cdot \frac{T.N_c}{\sum_{c \in C} T.N_c}$
21:         end for
22:     else
23:         $T_i \leftarrow \text{subtree}_i(T)$
24:         $\forall T_i$ : BottomUpAggregate($T_i$)
25:         for all $c \in C$ do
26:             $T.N_c \leftarrow \sum_i T_i.N_c$
27:         end for
28:     end if
29: end procedure
```

4.3 Continuous Attributes

One important extension that C4.5 added on top of ID3 was the ability to handle continuous attributes. Attribute values that appear in the learning examples are used to determine potential split points, which are then evaluated with the splitting criterion. Unfortunately, when inducing decision trees with differential privacy, it is not possible to use attribute values from the learning examples as splitting points; this would be a direct violation of privacy, revealing at the least information about the record that supplied the value of the splitting point.

The exponential mechanism gives us a different way to determine a split point: the learning examples induce a probability distribution over the attribute domain; given a splitting criterion, split points with better scores will have higher probability to be picked. Here, however, the exponential mechanism is applied differently than in Section 4.1: the output domain is not discrete. Fortunately, the learning examples divide the domain into ranges of points that have the same score, allowing for efficient application of the mechanism. The splitting point is sampled in two phases: first, the domain is divided into ranges where the score is constant (using the learning examples). Each range is considered a discrete option, and the exponential mechanism is applied to choose a range. Then, a point from the range is sampled with uniform distribution and returned as the output of the exponential mechanism. The probability assigned to the range in the first stage takes into account also the sampling in the second stage. This probability is obtained by integrating the density function induced by the exponential mechanism over the range. For example, consider a continuous attribute over the domain $[a, b]$. Given a dataset $d \in \mathcal{D}^n$ and a splitting criterion q, assume that all the points in $r \in [a', b']$ have the same score: $q(d, r) = c$. In that case, the exponential mechanism should choose this range with probability

$$\frac{\int_{a'}^{b'} \exp(\epsilon q(d,r)/2S(q))dr}{\int_a^b \exp(\epsilon q(d,r)/2S(q))dr} = \frac{\exp(\epsilon \cdot c) \cdot (b' - a')}{\int_a^b \exp(\epsilon q(d,r))dr} .$$

In general, given the ranges $R_1, \ldots, R_m$, where all the points in range R_i get the score c_i, the exponential mechanism sets the probability of choosing range R_i to be $\frac{\exp(\epsilon \cdot c_i) \cdot |R_i|}{\sum_i \exp(\epsilon \cdot c_i) \cdot |R_i|}$, where $|R_i|$ is the size of range R_i. Note that this approach is applicable only if the domain of the attribute in question is finite, and in our experiments we define the domain for each attribute in advance. This range cannot be determined dynamically according to the values observed in the learning examples, as this would violate differential privacy.

A split point should be determined for every numeric attribute. In addition, this calculation should be repeated for every node in the decision tree (after each split, every child node gets a different set of instances that require different split points). Therefore, supporting numeric attributes requires setting aside a privacy budget for determining the split points. To this end, given n numeric attributes, the budget distribution in line 3 of algorithm 2 should be updated to $\epsilon = \frac{B}{(2+n)d+2}$, and the exponential mechanism should be applied to determine a split point for each numeric attribute before line 14. An alternative solution is to discretize the numeric attributes before applying the decision tree induction algorithm, losing information in the process in exchange for budget savings.

5. EXPERIMENTAL EVALUATION

In this section we evaluate the proposed algorithms using synthetic and real data sets. The experiments were executed on Weka [22], an open source machine learning software. Since Weka is a Java-based environment, we did not rely on the PINQ framework, but rather wrote our own differential privacy wrapper to the `Instances` class of Weka, which holds the raw data. The algorithms were implemented on top of that wrapper. We refer to the implementation of algorithm 2 as *DiffPID3*, and to its extension that supports continuous attributes and pruning as *DiffPC4.5*.

5.1 Synthetic Datasets

We generated synthetic datasets using a method adapted from [6]. We define a domain with ten nominal attributes and a class attribute. At first, we randomly generate a decision tree up to a predetermined depth. The attributes are picked uniformly without repeating nominal attributes on the path from the root to the leaf. Starting from the third level of the tree, with probability $p_{\text{leaf}} = 0.3$ we turn nodes to leaves. The class for each leaf is sampled uniformly. In the second stage, we sample points with uniform distribution and classify them with the generated tree. Optionally, we introduce noise to the samples by reassigning attributes and classes, replacing each value with probability p_{noise} (we used $p_{\text{noise}} \in \{0, 0.1, 0.2\}$). The replacement is chosen uniformly, possibly repeating the original value. For testing, we generated similarly a noiseless test set with $10,000$ records.

5.1.1 *Comparing Splitting Criteria Over a Single Split*

In the first experiment we isolated a split on a single node. We created a tree with a single split (depth 1), with ten binary attributes and a binary class, which takes a different value in each leaf. We set the privacy budget to $B = 0.1$, and by varying the size of the training set, we evaluated the success of each splitting criterion in finding the correct split. We generated training sets with sizes ranging from 100 to 5000, setting 5000 as a bound on the dataset size for determining information gain sensitivity. For each sample size we executed 200 runs, generating a new training set for each, and averaged the results over the runs. Figure 3 presents the results for $p_{noise} = 0.1$, and Table 1 shows the accuracy and standard deviations for some of the sample sizes. In general, the average accuracy of the resulting decision tree is higher as more training samples are available, reducing the influence of the differential privacy noise on the outcome. Due to the noisy process that generates the classifiers, their accuracy varies greatly. However, as can be seen in the cases of the Max scorer and the Gini scorer, the influence of the noise weakens and the variance decreases as the number of samples grows. For the SuLQ-based algorithm,

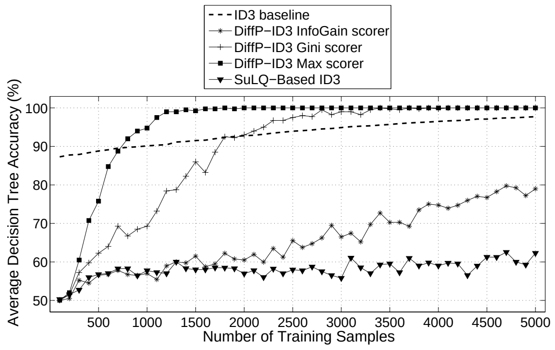

Figure 3: Splitting a single node with a binary attribute, $B = 0.1$, $p_{\text{noise}} = 0.1$

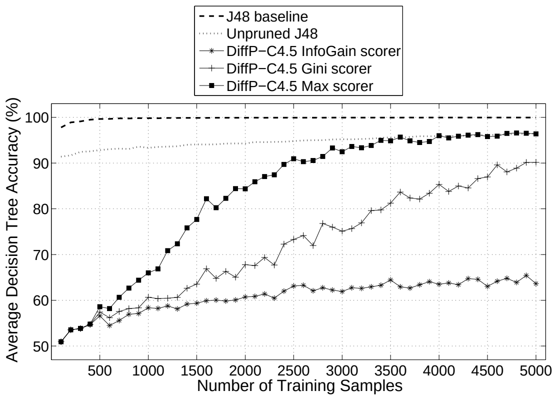

Figure 4: Splitting a single node with a continuous attribute, $B = 0.1$, $p_{\text{noise}} = 0.1$

when evaluating the counts for information gain, the budget per query is a mere 0.00125, requiring Laplace noise with standard deviation over 1000 in each query. On the other hand, the Max scorer, which is the least sensitive to noise, provides excellent accuracy even for sample sizes as low as 1500. Note that ID3 without any privacy constraints does not get perfect accuracy for small sample sizes, because it overfits the noisy learning samples.

Figure 4 presents the results of a similar experiment carried out with a numeric split. Three of the ten attributes were replaced with numeric attributes over the domain $[0, 100]$, and one of them was used to split the tree with a split point placed at the value 35. The results of the J48 algorithm (Weka's version of C4.5 v8) are provided as a benchmark, with and without pruning. The relation between the different scores is similar for the numeric and the nominal case, although more samples are required to correctly identify the split point given the privacy budget.

In Figure 5 we compare the performance of DiffPC4.5 on a numeric split as opposed to discretizing the dataset and running DiffPID3. The dataset from the former experiment was discretized by dividing the range of each numeric attribute to 10 equal bins. Of course, if the split point of a numeric attribute happens to match the division created by the discrete domains, working with nominal attributes would be

Number of samples	ID3	DiffPID3-InfoGain	DiffPID3-Gini	DiffPID3-Max	SuLQ-based ID3
1000	90.1 ± 1.5	57.0 ± 17.3	69.3 ± 24.3	94.7 ± 15.3	57.7 ± 18.1
2000	92.8 ± 1.0	60.5 ± 20.4	93.0 ± 17.4	100 ± 0.0	57.0 ± 17.4
3000	94.9 ± 0.7	66.1 ± 23.5	99.0 ± 7.0	100 ± 0.0	55.8 ± 15.9
4000	96.5 ± 0.6	74.7 ± 25.0	99.75 ± 3.5	100 ± 0.0	59.0 ± 19.2
5000	97.7 ± 0.5	79.0 ± 24.7	100 ± 0.0	100 ± 0.0	62.3 ± 21.5

Table 1: Accuracy and standard deviation of single split with binary attribute and binary class, $B = 0.1$

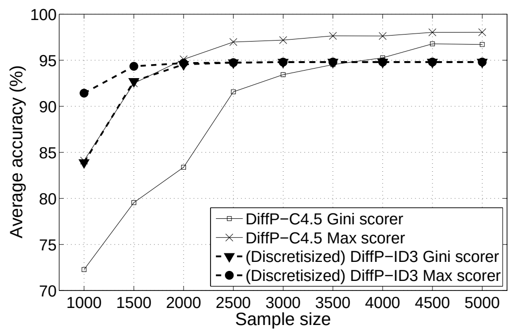

Figure 5: Comparing numerical split to discretization, $B = 0.2$, $p_{\text{noise}} = 0.0$

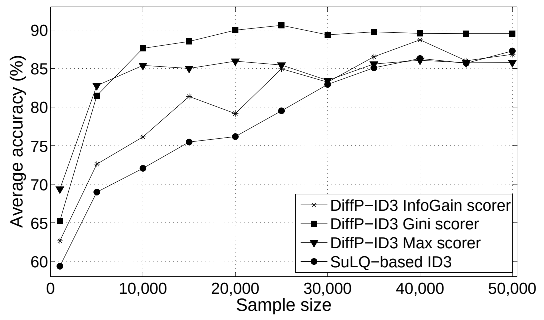

Figure 6: Inducing a tree of depth 5 with 10 binary attributes and a binary class, $B = 1.0$, $p_{\text{leaf}} = 0.3$

preferable, because determining split points for numeric attributes consumes some of the privacy budget. However, we intentionally used discrete domains that mismatch the split point (placed at the value 35), to reflect the risk of reduced accuracy when turning numeric attributes to discrete ones. The results show that for smaller training sets, the budget saved by discretizing the dataset and switching to nominal attributes allows for better accuracy. For larger training sets, the exponential mechanism allows, on average, split points to be determined better than in the discrete case.

5.1.2 Inducing a Larger Tree

We conducted numerous experiments, creating and learning trees of depths 3, 5 and 10, and setting the attributes and class to have 2, 3 or 5 distinct values. We used $B = 1.0$ over sample sizes ranging from 1000 to 50,000 instances, setting 50,000 as the bound on the dataset size for determining information gain sensitivity. For each tested combination of values we generated 10 trees, executed 20 runs on each tree (each on a newly generated training set), and averaged the results over all the runs. In general, the results exhibit behavior similar to that seen in the previous set of experiments. The variance in accuracy, albeit smaller than that observed for a single split, was apparent also when inducing deeper trees. For example, the typical standard deviation for the accuracy results presented in Figure 6 was around $\pm 5\%$ and even lower than that for the results presented in Figure 7. When inducing shallower trees or using attributes with fewer distinct values, we observed an interesting pattern, which is illustrated in Figures 6 and 7. When the size of the dataset is small, algorithms that make efficient use of the privacy budget are superior. This result is similar to the results observed in the previous experiments. However, as the number of available samples increases, the restrictions set by the privacy budget have less influence on the accuracy

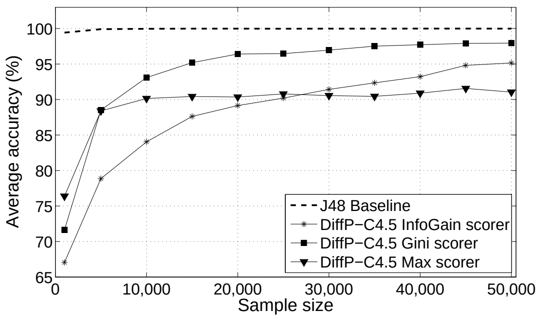

Figure 7: Inducing a tree of depth 5 with 7 binary attributes, 3 numeric attributes and a binary class, $B = 1.0$, $p_{\text{leaf}} = 0.3$

of the resulting classifier. When that happens, the depth constraint becomes more dominant; the Max scorer, which with no privacy restrictions usually produces deeper trees than those obtained with InfoGain or Gini scorers, provides inferior results with respect to the other methods when the depth constraint is present.

5.2 Real Dataset

We conducted experiments on the Adult dataset from the UCI Machine Learning Repository [5], which contains census data. The data set has 6 continuous attributes and 8 nominal attributes. The class attribute is income level, with two possible values, $\leq$50K or >50K. After removing records with missing values and merging the training and test sets, the dataset contains 45,222 learning samples (we set 50,000 as the bound on the dataset size for determining information gain sensitivity). We induced decision trees of depth up to 5 with varying privacy budgets. Note that although the values

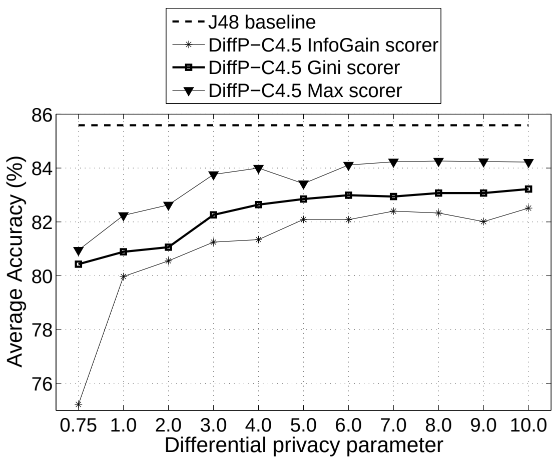

Figure 8: Accuracy vs. privacy (B) in the Adult dataset

considered for B should typically be smaller than 1, we used larger values to compensate for the relatively small dataset size[1] (the privacy budget toll incurred by the attributes is heavier than it was in the synthetic dataset experiments). We executed 10 runs of 10-fold cross-validation to evaluate the different scorers. Statistical significance was determined using corrected paired t-test with confidence 0.05. Figure 8 summarizes the results. In most of the runs, the Max scorer did significantly better than the Gini scorer, and both did significantly better than the InfoGain scorer. In addition, all scorers showed significant improvement in accuracy as the allocated privacy budget was increased. The typical measured standard deviation in accuracy was $\pm 0.5\%$.

6. CONCLUSIONS AND FUTURE WORK

We conclude that the introduction of formal privacy guarantees into a system requires the data miner to take a different approach to data mining algorithms. The sensitivity of the calculations becomes crucial to performance when differential privacy is applied. Such considerations are critical especially when the number of training samples is relatively small or the privacy constraints set by the data provider are very limiting. Our experiments demonstrated this tension between privacy, accuracy, and the dataset size. This work poses several future challenges. The large variance in the experimental results is clearly a problem, and more stable results are desirable even if they come at a cost. One solution might be to consider other stopping rules when splitting nodes, trading possible improvements in accuracy for increased stability. In addition, it may be fruitful to consider different tactics for budget distribution. Another interesting direction, following the approach presented in [14], is to relax the privacy requirements and allow ruling out rare calculation outcomes that lead to poor results.

[1]In comparison, some of the datasets used by previous works were larger by an order of magnitude or more (census data with millions of records [14], Netflix data of 480K users [16], search queries of hundreds of thousands users [15]).

7. REFERENCES

[1] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *Proc. of PODS*, pages 128–138, New York, NY, June 2005.

[2] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proc. of STOC*, pages 609–618, 2008.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Chapman & Hall, New York, 1984.

[4] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[5] C. B. D.J. Newman, S. Hettich and C. Merz. UCI repository of machine learning databases, 1998.

[6] P. Domingos and G. Hulten. Mining high-speed data streams. In *KDD*, pages 71–80, 2000.

[7] C. Dwork. Differential privacy. In *ICALP (2)*, volume 4052 of *LNCS*, pages 1–12, 2006.

[8] C. Dwork. Differential privacy: A survey of results. In *TAMC*, pages 1–19, 2008.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[10] C. Dwork and S. Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In *CRYPTO*, pages 469–480, 2008.

[11] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. In *STOC*, pages 361–370, 2009.

[12] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.

[13] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

[14] A. Machanavajjhala, D. Kifer, J. M. Abowd, J. Gehrke, and L. Vilhuber. Privacy: Theory meets practice on the map. In *ICDE*, pages 277–286, 2008.

[15] F. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *SIGMOD Conference*, pages 19–30, 2009.

[16] F. McSherry and I. Mironov. Differentially private recommender systems: building privacy into the net. In *KDD*, pages 627–636, 2009.

[17] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[18] J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4):319–342, 1989.

[19] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[20] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, 1993.

[21] R. E. Steur. *Multiple criteria optimization: theory computation and application.* John Wiley & Sons, New York, 1986.

[22] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, San Francisco, 2nd edition, 2005.

APPENDIX

A. SENSITIVITY OF SPLITTING CRITERIA

In this section we show how the sensitivity of the quality functions presented in Section 4.1 was derived.

A.1 Information gain

To evaluate the sensitivity of q_{IG}, we will use the following property:

Claim 1. For $a > 0$:

$$\left| a \log \frac{a+1}{a} \right| \leq \frac{1}{\ln 2} \ .$$

PROOF. The function $\left| a \log \frac{a+1}{a} \right|$ does not have extremum points in the range $a > 0$. At the limits, $\lim_{a \to \infty} a \log \frac{a+1}{a} = \frac{1}{\ln 2}$ and $\lim_{a \to 0} a \log \frac{a+1}{a} = 0$ (L'Hospital). $\square$

Let $r = \{r_1, r_2, \ldots, r_d\}$ be a record, let $\mathcal{T}$ be some dataset, and $\mathcal{T}' = \mathcal{T} \cup \{r\}$. Note that in the calculation of $q_{\mathrm{IG}}(\mathcal{T}, A)$ over the two datasets, the only elements in the sum that will differ are those that relate to the attribute $j \in A$ that the record r takes on the attribute A. Therefore, given this j we can focus on:

$$\begin{aligned} {q'}_{\mathrm{IG}}(\mathcal{T}, A_j) &= \sum_{c \in C} \tau_{j,c}^A \cdot \log \frac{\tau_{j,c}^A}{\tau_j^A} \\ &= \sum_{c \in C} \left(\tau_{j,c}^A \log \tau_{j,c}^A \right) - \log \tau_j^A \sum_{c \in C} \tau_{j,c}^A \\ &= \sum_{c \in C} \left(\tau_{j,c}^A \log \tau_{j,c}^A \right) - \tau_j^A \log \tau_j^A \ . \end{aligned}$$

The addition of a new record r affects only one of the elements in the left-hand sum (specifically, one of the elements $\tau_{j,c}^A$ increases by 1), and in addition, τ_j^A increases by one as well.

Hence we get that:

$$\begin{aligned} S(q_{\mathrm{IG}}) \leq & \Big| (\tau_{j,c}^A + 1) \log (\tau_{j,c}^A + 1) - \tau_{j,c}^A \log \tau_{j,c}^A + \\ & + \tau_j^A \log \tau_j^A - (\tau_j^A + 1) \log (\tau_j^A + 1) \Big| = \\ = & \left| \tau_{j,c}^A \log \frac{\tau_{j,c}^A + 1}{\tau_{j,c}^A} + \log (\tau_{j,c}^A + 1) - \right. \\ & \left. - \tau_j^A \log \frac{\tau_j^A + 1}{\tau_j^A} - \log (\tau_j^A + 1) \right| \ . \end{aligned}$$

The expressions $\tau_{j,c}^A \log \frac{\tau_{j,c}^A + 1}{\tau_{j,c}^A}$ and $\tau_j^A \log \frac{\tau_j^A + 1}{\tau_j^A}$ are both of the form $a \log \frac{a+1}{a}$. Therefore, we can apply Claim 1 and get that for a node with up to τ elements,

$$S(q_{\mathrm{IG}}) \leq \log(\tau + 1) + 1/\ln 2 \ .$$

Bounding the sensitivity of $q_{\mathrm{IG}}(\mathcal{T}, A)$ requires an upper bound on the total number of training examples. In our experiments, we assume that such a bound is given by the data provider. An alternate approach is to evaluate the number of training examples with a `NoisyCount` before invoking the exponential mechanism. The downside of this approach is that negative noise may provide a value for τ which is too small, resulting in insufficient noise. Therefore, in this approach there is a small chance that the algorithm would violate ϵ-differential privacy.

A.2 Gini index

Taking $p(j|t)$ to be the fraction of records that take the value t on attribute A and the class t, the Gini index can be expressed as $\mathrm{Gini} = \sum_{j \neq i} p(j|t) p(i|t) = 1 - \sum_j p^2(j|t)$. When determining the Gini index for a tree, this metric is summed over all the leaves, where each leaf is weighted according to the number of records in it. To minimize the Gini index, we observe that:

$$\begin{aligned} \min \mathrm{Gini}(A) &= \min \sum_{j \in A} \frac{\tau_j^A}{\tau} \left(1 - \sum_{c \in C} \left(\frac{\tau_{j,c}^A}{\tau_j^A} \right)^2 \right) \\ &= \min \sum_{j \in A} \tau_j^A \left(1 - \sum_{c \in C} \left(\frac{\tau_{j,c}^A}{\tau_j^A} \right)^2 \right) \ . \end{aligned}$$

As in information gain, the only elements in the expression that change when we alter a record are those that relate to the attribute value $j \in A$ that an added record r takes on attribute A. So, for this given j, we can focus on:

$$\begin{aligned} q'_{\mathrm{Gini}}(\mathcal{T}, A) &= \tau_j^A \left(1 - \sum_{c \in C} \left(\frac{\tau_{j,c}^A}{\tau_j^A} \right)^2 \right) \\ &= \tau_j^A - \frac{1}{\tau_j^A} \sum_{c \in C} \left(\tau_{j,c}^A \right)^2 \ . \end{aligned}$$

We get that:

$$\begin{aligned} S(q_{\mathrm{Gini}}(\mathcal{T}, A)) \leq & \left| (\tau_j^A + 1) - \frac{\left(\tau_{j,c_r}^A + 1 \right)^2 + \sum_{c \neq c_r} \left(\tau_{j,c}^A \right)^2}{(\tau_j^A + 1)} - \right. \\ & \left. - \tau_j^A + \frac{1}{\tau_j^A} \sum_{c \in C} \left(\tau_{j,c}^A \right)^2 \right| = \\ = & \left| 1 + \frac{(\tau_j^A + 1) \sum_{c \in C} \left(\tau_{j,c}^A \right)^2}{\tau_j^A (\tau_j^A + 1)} - \right. \\ & \left. - \frac{\tau_j^A \left(\left(\tau_{j,c_r}^A + 1 \right)^2 + \sum_{c \neq c_r} \left(\tau_{j,c}^A \right)^2 \right)}{\tau_j^A (\tau_j^A + 1)} \right| = \\ = & \left| 1 + \frac{\sum_{c \in c} \left(\tau_{j,c}^A \right)^2 - \tau_j^A \left(2 \tau_{j,c_r}^A + 1 \right)}{\tau_j^A (\tau_j^A + 1)} \right| = \\ = & \left| 1 + \frac{\sum_{c \in C} \left(\tau_{j,c}^A \right)^2}{\tau_j^A (\tau_j^A + 1)} - \frac{\left(2 \tau_{j,c_r}^A + 1 \right)}{\tau_j^A + 1} \right| \ . \end{aligned}$$

In the last line, $0 \leq \left| \frac{\sum_{c \in C} \left(\tau_{j,c}^A \right)^2}{\tau_j^A (\tau_j^A + 1)} \right| \leq \left| \frac{(\tau_j^A)^2}{\tau_j^A (\tau_j^A + 1)} \right| \leq 1$ due to $\sum \tau_{j,c}^A = \tau_j^A$ and the triangle inequality. In addition, since $\tau_{j,c}^A \leq \tau_j^A$, we get that $0 \leq \frac{2\tau_{j,c_r}^A + 1}{\tau_j^A + 1} \leq \frac{2\tau_j^A + 1}{\tau_j^A + 1} \leq 2$. Therefore,

$$S(q_{\mathrm{Gini}}) \leq 2 \ .$$

A.3 Max

This query function is adapted from the resubstitution estimate described in [3]. In a given tree node, we would like to choose the attribute that minimizes the probability of misclassification. This can be done by choosing the attribute that maximizes the total number of hits. Since a record can change the count only by 1, we get that $S(q_{\mathrm{Max}}) = 1$.