Improving Private Random Forest Prediction Using Matrix Representation

Arisa Tajima[1], **Joie Wu**[2], **Amir Houmansadr**[1]
[1]University of Massachusetts Amherst
[2]Independent Researcher
atajima@cs.umass.edu, joie.y.wu@gmail.com, amir@cs.umass.edu

Abstract

We introduce a novel matrix representation for differentially private training and prediction methods tailored to random forest classifiers. Our approach involves representing each root-to-leaf decision path in all trees as a row vector in a matrix. Similarly, inference queries are represented as a matrix. This representation enables us to collectively analyze privacy across multiple trees and inference queries, resulting in optimal DP noise allocation under the Laplace Mechanism. Our experimental results show significant accuracy improvements of up to 40% compared to state-of-the-art methods.

Introduction

Recent advances in machine learning services empower users to make inference queries using deployed models through black-box APIs. However, the outcomes of these predictions raise concerns about potential information disclosure regarding the training data, making them vulnerable to attacks like membership inference attacks (Shokri et al. 2017; Hu et al. 2022). To address this, differential privacy (DP) (Dwork and Lei 2009) has emerged as a promising solution, aiming to mitigate the risk of model predictions revealing sensitive information.

Two primary DP-based approaches exist for private prediction: *DP training* and *DP prediction* (Ponomareva et al. 2023). DP training (Chaudhuri, Monteleoni, and Sarwate 2011; Jayaraman and Evans 2019; Abadi et al. 2016; Fletcher and Islam 2019) integrates DP noise into model parameters, ensuring that predictions from privately trained models do not reveal information about the underlying training data. By contrast, DP prediction (Dwork and Feldman 2018; Nissim, Raskhodnikova, and Smith 2007; Bassily, Thakkar, and Guha Thakurta 2018) introduces perturbations to the predictions of non-private models. Unfortunately, despite the widespread adoption of DP in machine learning, a significant utility-privacy gap persists between DP-enabled machine learning and its non-private counterparts.

DP-enabled Random Forests: In this paper, we focus on random forest classifiers—a versatile machine learning algorithm known for its strong performance with tabular data like demographic surveys and medical records. We address the challenge of private prediction through both DP training and prediction approaches while maintaining high accuracy.

DP random forest classifiers have been extensively studied. Past approaches primarily differ in their use of base tree algorithms and node splitting functions to balance privacy budget usage and accuracy (Patil and Singh 2014; Jagannathan, Pillaipakkamnatt, and Wright 2009; Fletcher and Islam 2017, 2019; Hou et al. 2019). Regarding the tree-building process, there are greedy approaches, where optimal split points are determined (Hou et al. 2019; Patil and Singh 2014) and random approaches, where split attributes are chosen randomly to save privacy budget for leaf nodes (Fletcher and Islam 2017; Jagannathan, Pillaipakkamnatt, and Wright 2009; Holohan et al. 2019). The latter method, which we focus on, is known as *random decision trees*. It is predominantly adapted in recent works for its consistently strong performance; see the comprehensive survey by Fletcher et al. (Fletcher and Islam 2019).

While distributing the privacy budget is crucial for DP random forests, many approaches allocate it equally among trees (Patil and Singh 2014; Jagannathan, Pillaipakkamnatt, and Wright 2009; Fletcher and Islam 2017; Hou et al. 2019). However, this per-tree privacy analysis faces challenges when a large number of trees are used, resulting in low accuracy despite the ensemble learning principle that training more trees generally improves performance.

To our knowledge, no prior work has addressed DP *prediction* techniques specifically for random forests. Alternatively, a well-known model-agnostic DP prediction method—the subsample-and-aggregate framework—allocates the privacy budget equally to each inference query (Dwork and Feldman 2018; Nissim, Raskhodnikova, and Smith 2007). However, this heuristic allocation results in poor accuracy when many inference queries are required (van der Maaten and Hannun 2020).

Main Contributions

Existing methods for DP training and DP prediction add suboptimal levels of noise, hampering utility. Our primary contribution lies in addressing the inefficiencies of current methods *by translating random forest training and prediction into matrix multiplication*. This new representation allows us to optimize solutions for training models and predicting class labels under DP, introducing finer-grained noise than the state-of-the-art.

We introduce *DP batch training*, which allocates the privacy budget across an ensemble of trees. The key insight behind our approach is the observation that some leaf values can be expressed as linear combinations of others. We optimize budget allocation based on leaf values along decision paths, preserving high accuracy even as the number of trees learned increases. Our allocation strategy is data-independent, unlike a recent method that uses weighted privacy budget allocation for trees, which incurs budget expenditure (Li et al. 2022). Thus, our approaches offer privacy-free hyperparameter tuning, a significant advantage over existing methods including DP-SGD (Abadi et al. 2016).

Additionally, we introduce *DP batch prediction* which takes into account prediction results collectively rather than isolating individual queries. This technique optimizes DP noise addition across a set of inference queries for majority voting in an ensemble. Batch privacy analysis closes the performance gap between DP prediction performance and DP training due to the former's limit on inference queries. Our results show that this approach maintains good accuracy, even as the number of inference queries increases.

We validate our methods on real-world datasets, demonstrating a significant accuracy improvement of up to 40% compared to existing approaches. When tested on our Car dataset, both of our DP batch training and DP batch prediction approaches achieve 85% accuracy under a privacy budget of $\epsilon = 2$ when predicting 345 test samples on 128 random decision trees, exceeding the accuracy of existing solutions by 30%. Finally, our matrix representation can be extended to work within the subsample-and-aggregate framework, yielding an improvement of up to 50%. Our code and technical appendix will be available at: https://github.com/arisa77/mrf-public.git.

Background

Data and Schema

Schema. Consider a schema consisting of d attributes: $\{A_1, A_2, \ldots, A_d\}$. Each attribute A_i has a finite domain $\Phi(A_i)$ of size n_i. The full domain size is then $\prod_{i=1}^{d} n_i$. For clarity, we assume the first $d-1$ attributes represent feature attributes with domains denoted as $\mathcal{X}_i$ for $i \in [d-1]$. Let $\mathcal{X} = \prod_{i=1}^{d-1} \mathcal{X}_i$, and $n = \prod_{i=1}^{d-1} n_i$. The last attribute is the target class attribute, denoted as $\mathcal{Y}$ with k variables. We distinguish $\mathcal{X}$ as the d-1 dimensional feature space and $\mathcal{Y}$ as the 1-dimensional target space.

Example 1 (Tennis Schema) *We use a simplified tennis dataset with attributes* $\{outlook, windy, play\}$ *and domains* $\Phi(outlook) = \{sunny, overcast, rainy\}$, $\Phi(windy) = \{false, true\}$, *and* $\Phi(play) = \{no, yes\}$. *The features are 'outlook' and 'windy', and the target is 'play'. The feature domain* $\mathcal{X}$ *contains all tuples from* $\Phi(outlook) \times \Phi(windy)$, *resulting in* $\mathcal{X} = \{$*(sunny, false), (sunny, true), (overcast, false), (overcast, true), (rainy, false), (rainy, true)*$\}$. *The target domain is* $\mathcal{Y} = \{no, yes\}$.

Data Matrix. We have a sensitive training dataset consisting of N individuals, represented as an $N \times d$ matrix X. Each row X_i is an individual record and $X_{i,j} \in \Phi(A_j)$ is the value of attribute A_j. We often represent this dataset X as a 2-way contingency table of feature variables $\mathcal{X}$ by target variables $\mathcal{Y}$, denoted as a matrix $\mathbf{D}_X \in \mathbb{R}^{n \times k}$. We may drop the subscript X and write $\mathbf{D}$ if the context is clear. This matrix captures the frequency of every possible tuple $(x, y) \in \mathcal{X} \times \mathcal{Y}$ in X. Although the frequency representation is favored for mathematical convenience, our implementation uses the record-by-record format for efficiency.

Example 2 (Tennis Dataset) *Consider the tennis dataset with six samples shown in Figure 1a. The corresponding 6 by 2 frequency matrix* $\mathbf{D}$ *is illustrated in Figure 1c. For instance,* $\mathbf{D}_{1,1} = 1$ *indicates a single sample with features (sunny, false) and a target class of no.*

Differential Privacy

Differential privacy (DP) (Dwork, Roth et al. 2014) is the defacto standard for data privacy, formally defined as follows.

Definition 1 (Differential Privacy) *A randomized mechanism* $\mathcal{M}$ *satisfies* ϵ*-DP if for any two neighboring datasets* $X, X' \in \mathcal{D}$ *and for any subset of outputs* $\mathcal{S} \subseteq Range(\mathcal{M})$ *it holds that:* $\Pr[\mathcal{M}(X) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(X') \in \mathcal{S}]$.

DP offers valuable properties such as the post-processing and composition theorem (Dwork, Roth et al. 2014), which we utilize in this work. Following the literature on DP random forests, we focus on employing the Laplace mechanism (Fletcher and Islam 2019; Holohan et al. 2019).

Matrix Mechanism. Matrix Mechanism (Li et al. 2015) is a variant of the Laplace mechanism designed for answering input queries defined by matrix $\mathbf{W}$ by using a set of underlying queries $\mathbf{A}$ such that there exists some $\mathbf{X}$ for which $\mathbf{W} = \mathbf{XA}$. $\mathbf{A}$ is referred to as the *strategy matrix*. The naive case where $\mathbf{A} = \mathbf{W}$ corresponds to the vectorized Laplace mechanism. Heuristics for choosing the best strategy matrix have been widely studied (Xiao, Gardner, and Xiong 2012; Xiao, Wang, and Gehrke 2010).

Definition 2 (Matrix Mechanism) *Given* l *by* n *data-independent query matrix* $\mathbf{W}$, m *by* n *data-independent strategy matrix* $\mathbf{A}$ *(possibly induced by* $\mathbf{W}$*), and* n *by* k *data matrix* $\mathbf{D}_X$*, the following Matrix Mechanism satisfies* ϵ*-DP.* $\mathcal{M}_{\mathbf{A}}(\mathbf{W}, \mathbf{D}_X) = \mathbf{W}\mathbf{D}_X + \mathbf{W}\mathbf{A}^{+}\text{Lap}(||\mathbf{A}||_1/\epsilon)^{m \times k}$.

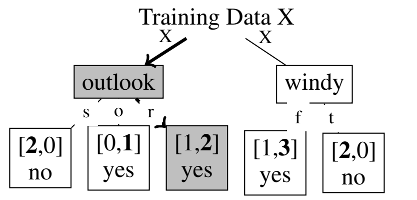
The sensitivity of a query matrix is defined as its L1 norm $||\mathbf{A}||_1$. $\mathbf{A}^{+}$ denotes the pseudoinverse of $\mathbf{A}$. The original work utilizes a data vector instead of a data matrix, although both expressions are essentially interchangeable. The mean-squared error of query answers to $\mathbf{W}$ under the strategy matrix $\mathbf{A}$ is given as follows, independent of the actual data input. $||\cdot||_F$ denotes the Frobenius norm.

Definition 3 (Error of strategy query answering) *Given query matrix* $\mathbf{W}$, *strategy matrix* $\mathbf{A}$, *the total mean squared error of Matrix Mechanism is given as:* $\text{Err}_{\epsilon}(\mathbf{W}, \mathbf{A}) = 2/\epsilon^2 ||\mathbf{A}||_1^2 ||\mathbf{W}\mathbf{A}^{+}||_F^2$.

The state-of-the-art minimizes this error through parameterized optimization, identifying a $(p+n) \times n$ strategy matrix $\mathbf{A}$ by finding a $p \times n$ parameter matrix (McKenna et al. 2018). We treat the optimization routine as a black box, denoted as $\mathbf{A} \leftarrow \text{OPT}_p(\mathbf{W})$, where p is a hyperparameter.

outlook	windy	play
sunny	false	no
sunny	true	no
overcast	false	yes
rainy	false	yes
rainy	false	yes
rainy	true	no

(a) Training data



(b) Random decision trees with $\tau = 2, h = 1$

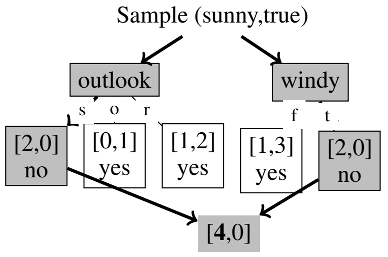
$$\overbrace{\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}}^{\text{feature domain}}_{\mathbf{T}\in\mathbb{R}^{o\times n}} \underset{\mathbf{D}\in\mathbb{R}^{n\times k}}{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}} = \underset{\mathbf{C}\in\mathbb{R}^{o\times k}}{\begin{bmatrix} \mathbf{2} & 0 \\ 0 & \mathbf{1} \\ 1 & \mathbf{2} \\ 1 & \mathbf{3} \\ \mathbf{2} & 0 \end{bmatrix}}$$

(c) Computing leaf values

outlook	windy	play
sunny	**true**	-
overcast	true	-
overcast	false	-

$$\underset{\mathbf{Q}\in\mathbb{R}^{b\times n}}{\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}$$

(d) Test data and corresponding query $\mathbf{Q}$



(e) Weight voting for prediction

$$\overbrace{\begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 0 & 1 & 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 & 1 & 0 \end{bmatrix}}^{\text{feature domain}}_{\mathbf{Q}\mathbf{T}^{\intercal}\mathbf{T}\in\mathbb{R}^{b\times n}} \overbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}}^{\text{target domain}}_{\mathbf{D}\in\mathbb{R}^{n\times k}} = \underset{\mathbf{V}\in\mathbb{R}^{b\times k}}{\begin{bmatrix} \mathbf{4} & 0 \\ \mathbf{2} & 1 \\ 1 & \mathbf{4} \end{bmatrix}}$$

(f) Aggregating weighted vote counts

Figure 1: Matrix multiplication for computing leaf values for training (c) and weighted voting for prediction (d) in random decision trees. The matrices $\mathbf{D}$, $\mathbf{T}$, and $\mathbf{Q}$ correspond to the training data (a), tree decision paths (b), and test samples (d).

Random Decision Trees

Random decision tree models were initially proposed as efficient classifiers for large databases in non-private settings (Fan et al. 2003; Geurts, Ernst, and Wehenkel 2006). They have since become a dominant algorithm for state-of-the-art DP random forests due to their data-independent tree construction (Jagannathan, Pillaipakkamnatt, and Wright 2009; Holohan et al. 2019; Fletcher and Islam 2017). Unlike traditional decision trees such as ID3 and CART, random decision trees are constructed by randomly selecting test attributes for nodes. Because this attribute selection occurs before examining any training data, there is no need to allocate a privacy budget for finding optimal split points.

Base Classifiers. We consider τ random decision trees each with depth h, denoted as $F = \{T_1, \dots, T_\tau\}$. We may denote F_S as those built over a specific feature subset S. We may build q ensembles of such random decision trees, each associated with a random feature subset, denoted as $F = \{F_{S_j}\}_{j=1}^{q}$. The tree structure, including random feature selection, is pre-computed using schema information alone. Training data X is then used to calculate class count distributions at all leaves. Each tree is fitted on the same training dataset unless stated otherwise. We provide the full algorithm of random decision trees in the technical appendix. During prediction, the trained random decision trees F take a sample $s \in \mathcal{X}$ and output a predicted label $y \in \mathcal{Y}$, denoted as $y \leftarrow F(s)$. Each tree casts a vote, and the label with the most votes is chosen as the final prediction.

Private Prediction and Problem Statement. Given b inference queries $Q \in \mathcal{X}^b$, our goal is to release prediction results $\mathbf{y} \leftarrow F(Q)$ (abusing the notation above) under DP to limit information leakage about the training data X from the predictions. We will introduce formulations for both *DP training* and *DP prediction*. DP training is achieved by making the random forest F DP. Thus, the subsequent prediction algorithm maintains privacy due to the post-processing theorem. DP prediction, on the other hand, is achieved by adding DP noise to the predictions of the non-DP random forest F.

Matrix Random Forests

We introduce a novel approach to representing random forest training and prediction using matrices, which we call *Matrix Random Forest (M-RF)*. This approach is essential in generating the optimal amount of DP noise, as we will detail later.

Inference Query Matrix

A sample $s \in \mathcal{X}$ is represented as an indicator vector $\mathbf{q}$ of length n over $\mathcal{X}$, where the cell at the corresponding tuple is set to 1. Otherwise, it is set to 0. Thus, a set of b inference queries $\{s_1, \dots, s_b\} \in \mathcal{X}^b$ can be expressed as a $b \times n$ matrix $\mathbf{Q} = (\mathbf{q}_1^\intercal, \dots, \mathbf{q}_b^\intercal)$. In contrast to the data matrix $\mathbf{D}$, the inference query matrix $\mathbf{Q}$ is *not* treated as sensitive.

Example 3 (Inference Query) *Figure 1d shows three unlabeled samples and the corresponding* 3×6 *inference query matrix* $\mathbf{Q}$. *For instance, the first row of* $\mathbf{Q}$ *encodes the tuple (sunny, true) from the dataset.*

Decision Path Query Matrix for Training

We introduce the decision path matrix that encodes every root-to-leaf decision path in random decision trees. Each decision path represents a predicate $P(x)$ over the feature domain $\mathcal{X}$. This predicate can be represented as a binary vector of length $n = |\mathcal{X}|$, denoted as $\mathbf{p}$, where $\mathbf{p}_i = 1$ if the corresponding element in $\mathcal{X}$ is in the truth set (the set of all elements in $\mathcal{X}$ that make $P(x)$ true); otherwise $\mathbf{p}_i = 0$. For instance, if $P(x)$ stands for "outlook is sunny" in Example 1, the truth set is $\{$(sunny, false), (sunny, true)$\}$ and thus the corresponding predicate results in $\mathbf{p} = (1, 1, 0, 0, 0, 0)$.

A predicate is used in a counting query to evaluate the number of instances in the dataset satisfying its corresponding decision path. For a single predicate $\mathbf{p}$, the expression $\mathbf{pD} \in \mathbb{R}^k$ will yield instance counts for each class label.

Thus, a set of o decision paths can be organized into an o by n matrix where each row corresponds to a predicate, denoted as $\mathbf{T} = (\mathbf{p}_1^\intercal, \ldots, \mathbf{p}_o^\intercal)$. The matrix encodes the *data-independent* tree structure. The class counts at every leaf node can be computed as $\mathbf{C} = \mathbf{TD} \in \mathbb{R}^{o\times k}$. Each $\mathbf{C}_{i,j}$ represents the frequency of data instances that reach the i-th leaf node whose class label is $y_j \in \mathcal{Y}$; see Figure 1c. We may write $\mathbf{C}_\mathbf{D}$ to emphasize its dependence on $\mathbf{D}$.

Decision Path Query Matrix for Prediction

For prediction, we construct the decision path matrix for individual instances. Given the decision path matrix $\mathbf{T} \in \mathbb{R}^{o\times n}$ and the inference query matrix $\mathbf{Q} \in \mathbb{R}^{b\times n}$ introduced earlier, their matrix multiplication $\mathbf{W} = \mathbf{QT}^\intercal \in \mathbb{R}^{b\times o}$ produces the specific decision paths for each sample. Specifically, $\mathbf{W}_{i,j} = 1$ if the i-th sample reaches the j-th leaf node on the j-th decision path defined by $\mathbf{T}_j$; otherwise, $\mathbf{W}_{i,j} = 0$. Thus, each row vector $\mathbf{W}_i$ encodes the decision paths taken by sample i across the random decision trees.

Weight Voting Matrix Operation

We introduce the matrix operations underlying the majority voting process for random decision trees. This method, which we call *weight voting*, involves aggregating leaf values across trees to make predictions. The aggregated votes represent a weighted count of data points that agree on the predicted class for each sample.

Given random decision trees, let $\mathbf{T}$ denote the decision path matrix and $\mathbf{C}_\mathbf{D}$ the leaf value matrix. For an inference query matrix $\mathbf{Q}$, the weight votes for each target class are calculated using the matrix product: $\mathbf{V} = \mathbf{QT}^\intercal\mathbf{C}_\mathbf{D} \in \mathbb{R}^{b\times k}$. Here $\mathbf{V}_{i,j}$ represents the number of training instances that agree with the class label $y_j \in \mathcal{Y}$ for the i-th inference query. When $\mathbf{C} = \mathbf{TD}$, this operation can be expressed in terms of $\mathbf{D}$ as: $\mathbf{V} = \mathbf{QT}^\intercal\mathbf{TD}$. Here, $\mathbf{QT}^\intercal\mathbf{T}$ represents weighted queries that aggregate tuples in the feature domain; see Figure 1f for an example matrix.

DP M-RF Training and Prediction

In this section, we present novel techniques for DP training and DP prediction of random forests. Our approach leverages the matrix representation formulated in the previous sections, enabling the derivation of optimal solutions that significantly improve the accuracy of DP random forests.

DP Batch Training Approach

We present our DP random forest training, referred to as *batch training*. This method provides DP leaf values while maintaining high accuracy by optimizing a batch of decision path queries. The leaf labels are computed from DP leaf values, ensuring the resultant random forests satisfy DP.

```
Algorithm 1: DP M-RF Training
Input: Data matrix $\mathbf{D}$, decision path matrix $\mathbf{T}$, privacy budget $\epsilon$.
Output: DP leaf label vector $\tilde{\mathbf{l}}$
1: $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{T})$
2: $\tilde{\mathbf{C}} = \mathbf{TD} + \mathbf{TA}^{+}\mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$
3: $\tilde{\mathbf{l}}_i = \arg\max_{1\leq j\leq k} \tilde{\mathbf{C}}_{i,j}, \forall$ leaf $i$.
```

Optimizing Decision Paths Utilizing our matrix representation, leaf values are computed as the matrix product $\mathbf{C} = \mathbf{TD}$, where $\mathbf{T}$ represents root-to-leaf paths in random decision trees. This serves as a query matrix, where each row vector counts the number of training data instances satisfying the classification rule of the corresponding leaf node. Many existing DP approaches apply the Laplace mechanism by adding the Laplace noise to the leaf values with an equal privacy budget allocation: $\mathbf{TD} + \mathrm{Lap}(\tau/\epsilon)^{o\times k}$, assigning each tree a budget of τ/ϵ. However, this method can lead to suboptimal accuracy if decision paths in one tree are linearly dependent on those in others, wasting the privacy budget.

To address this, we optimize leaf-level counts by finding an optimal strategy $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{T})$. By specifying the decision path query matrix $\mathbf{T}$ into the optimization procedure OPT, we derive an alternative strategy matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ that minimizes the error in estimating class counts at leaf nodes, i.e., Err$(\mathbf{T}, \mathbf{A})$. Under this strategy, ϵ-DP leaf values are estimated as $\tilde{\mathbf{C}} = \mathbf{TD} + \mathbf{TA}^{+}\mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$.

DP Matrix Random Forest Training Algorithm 1 presents our approach for training DP random forests. The decision path matrix $\mathbf{T}$ is precomputed by building random decision trees. With the data matrix $\mathbf{D}$ and the decision path matrix $\mathbf{T}$, we estimate leaf values under DP, following our above optimization approach. The resulting DP leaf values have the following error, a direct implication from Definition 3. Finally, the most frequent class is assigned as the label for each leaf node based on the noisy leaf values.

Theorem 4 *Leaf values $\tilde{\mathbf{C}}$ resulting from Algorithm 1 have MSE of* $\mathrm{Err}_\epsilon(\mathbf{T}, \mathbf{A})$.

Theorem 5 *Algorithm 1 satisfies ϵ-DP.*

The strategy matrix selection in Line 1 incurs no privacy loss since it only depends on the data-independent decision path matrix $\mathbf{T}$. The computation on leaf values in Line 2 satisfies ϵ-DP, following the privacy of the Matrix Mechanism. Finally, from the post-processing theorem, the computation of leaf labels in Line 3 does not degrade privacy.

DP Batch Prediction Approach

We introduce our DP random forest prediction approach, referred to as: *batch prediction*. Our approach predicts labels for given inference queries under DP while maintaining high accuracy. Unlike DP batch training, which optimizes leaf

Algorithm 2: DP M-RF Prediction

```
Input: Data matrix $\mathbf{D}$, decision path matrix $\mathbf{T}$, inference query matrix $\mathbf{Q}$, privacy budget $\epsilon$.
Output: DP predicted label vector $\tilde{\mathbf{y}}$
1: $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{Q}\mathbf{T}^\intercal\mathbf{T})$
2: $\tilde{\mathbf{V}} = \mathbf{Q}\mathbf{T}^\intercal\mathbf{T}\mathbf{D} + \mathbf{Q}\mathbf{T}^\intercal\mathbf{T}\mathbf{A}^{+}\mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$
3: $\tilde{\mathbf{y}}_i = \arg\max_{1\le j\le k} \tilde{\mathbf{V}}_{i,j}, \forall$ inference query $i$.
```

values for classifiers, DP batch prediction focuses on optimizing prediction results for specified inference queries.

Optimizing Decision Paths for Prediction The random forest prediction problem using weight voting can be expressed as the matrix product: $\mathbf{V} = \mathbf{Q}\mathbf{T}^\intercal\mathbf{T}\mathbf{D}$, where $\mathbf{Q}$ denotes the inference query matrix, and $\mathbf{T}$ represents decision paths in the trees. We consider $\mathbf{Q}\mathbf{T}^\intercal\mathbf{T}$ as a query matrix where each vector aggregates instance counts (i.e., weighted votes) at predicted leaf nodes across all trees for the corresponding inference query. Using the optimization procedure OPT, we find a $m \times n$ strategy matrix $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{W})$, where $\mathbf{W} = \mathbf{Q}\mathbf{T}^\intercal\mathbf{T}$, that minimizes the error of answering votes for inference queries, i.e., Err$(\mathbf{W}, \mathbf{A})$. Under this strategy, ϵ-DP vote counts are estimated as: $\tilde{\mathbf{V}} = \mathbf{W}\mathbf{D} + \mathbf{W}\mathbf{A}^{+}\mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$.

DP Matrix Random Forest Prediction Algorithm 2 shows our DP prediction approach for random forests. Given non-private random decision trees, the resulting tree structure are encoded into a decision path matrix $\mathbf{T}$. Training data and inference queries are similarly transformed into matrices $\mathbf{D}$ and $\mathbf{Q}$. We then estimate votes tallied across the forest for every inference query under DP, following the above optimization approach. The resulting DP vote counts have the following error, a direct implication from Definition 3. Lastly, class labels with the majority noisy votes are returned as the final prediction results.

Theorem 6 *The vote counts* $\mathbf{V}$ *in Algorithm 2 have MSE of* $\mathrm{Err}_\epsilon(\mathbf{Q}\mathbf{T}^\intercal\mathbf{T}, \mathbf{A})$.

Theorem 7 *Algorithm 2 satisfies ϵ-DP. Similar to the privacy proof of Theorem 5, the privacy of Algorithm 2 primarily follows from the privacy of the Matrix Mechanism.*

Optimizing Subsample-and-Aggregate Framework

We enhance the subsample-and-aggregate method (Dwork and Feldman 2018) using our batch prediction technique. In this approach, non-private random decision trees are fitted on *disjoint* training datasets to estimate weighted votes $\mathbf{V} = \mathbf{Q}\mathbf{T}^\intercal\mathbf{C}$ under DP. Since $\mathbf{C} \neq \mathbf{T}\mathbf{D}$, the above DP batch prediction is not directly applicable. To address this, our approach identifies an $m\times o$ strategy matrix $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{Q}\mathbf{T}^\intercal)$ that minimizes $\mathrm{Err}_\epsilon(\mathbf{Q}\mathbf{T}^\intercal, \mathbf{A})$. The DP vote counts are then computed as $\tilde{\mathbf{V}} = \mathbf{Q}\mathbf{T}^\intercal\mathbf{C} + \mathbf{Q}\mathbf{T}^\intercal\mathbf{A}^{+}\mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$. This method ensures query sensitivity remains independent of the number of inference queries, unlike the existing method, which allocates an equal budget among queries, significantly degrading accuracy with more queries. Further details appear in the technical appendix.

Our Full DP M-RF Framework

Our complete framework for private prediction with random decision trees is shown in Algorithm 3, capturing both DP batch training and DP batch prediction. The framework involves: 1) building trees, 2) optimizing strategies, 3) fitting training data, and 4) making predictions, detailed below.

We construct q ensembles of random decision trees, where each ensemble F_i is built over a random feature subset S_i of size $\bar{d}$. Our method involves performing strategy optimizations within each ensemble over the schema S_i, allowing potentially expensive computations to be performed in parallel across ensembles. The best strategy matrix $\mathbf{A}^{(i)}$ is chosen from OPT for generating refined DP noise. The query matrix $\mathbf{W}^{(i)}$ is instantiated with $\mathbf{T}^{(i)}$ or $\mathbf{Q}(\mathbf{T}^{(i)})^\intercal\mathbf{T}^{(i)}$, depending on batch training or batch prediction, recalling the prior sections. Here, $\mathbf{T}^{(i)}$ denotes the decision path matrix associated with ensemble i. The optimality of our approach comes from the theoretical optimality of the strategy matrix from OPT for a fixed strategy. The derived strategy matrix remains optimal within the constraints of each individual ensemble.

Private training data is fit on each ensemble forest by updating the leaf values and labels. For DP training, the refined DP noise is added to leaf values, and the resulting random decision trees $\tilde{F}_i$ are privatized. For DP prediction, the DP noise is added to the predicted vote counts, with predictions made on non-DP random decision trees. Finally, prediction labels are determined collectively across the q ensembles.

Algorithm 3 can capture our improved subsample-and-aggregate method as well by setting $\mathbf{W}^{(i)} = \mathbf{Q}(\mathbf{T}^{(i)})^\intercal$ and fitting *disjoint subsets* of data on each ensemble forest.

For privacy, following Theorem 5 and 7, each ensemble algorithm with a privacy budget of ϵ/q satisfies ϵ/q-DP for batch training and prediction. Thus, from the composition theorem, Algorithm 3 satisfies ϵ-DP.

Complexity. The primary overhead of DP M-RF in Algorithm 3 comes from the optimization for refining DP noise, crucial for balancing accuracy and efficiency. This process is parallelizable across ensembles for efficiency. Assuming each feature has n values with a total domain size of $n^{\bar{d}}$, the size of $\mathbf{W}^{(i)}$ is $o \times n^{\bar{d}}$, $b \times n^{\bar{d}}$, $b \times o$ for DP batch training, DP batch prediction, and subsample-and-aggregate. Here, o is the total number of leaves per ensemble and b is the number of inference queries. Our implementation adopts an implicit matrix representation, effectively reducing the matrix size to e.g., $o \times \bar{d}n$ for batch training (McKenna et al. 2018).

The optimization cost depends on the chosen strategies, producing a $(p_i + n^{\bar{d}}) \times n^{\bar{d}}$ matrix $\mathbf{A}^{(i)}$ with $p_i = O(n^{\bar{d}})$. The space and time complexity of noise generation in Line 6 are both $O(n^{\bar{d}})$ for DP batch training and prediction; for subsample-and-aggregate, they are $O(o+b)$ and $O((o+b)o)$, independent of domain sizes.

Tree building and optimization do not require the actual training data and can be preprocessed efficiently. Thus, the overhead for DP training and prediction is negligible. Moreover, matrices $\mathbf{Y}^{(i)}$ and $\mathbf{V}^{(i)}$ are computed without materializing the data matrix $\mathbf{D}$, ensuring efficient space complexity.

Algorithm 3: DP M-RF Framework

```
Input: features $S$, training data $X$, inference queries $Q$, privacy budget $\epsilon$, number of ensembles $q$, number of trees $\tau_1, \dots, \tau_q$, tree depth $h_1, \dots, h_q$, max feature size $\bar{d}$.
Output: DP predicted label vector $\tilde{\mathbf{y}}$
1: for every ensemble $i = 1 \dots q$ do
2:     $S_i \leftarrow \bar{d}$ random features from $S$
3:     $F_i \leftarrow \cup^{\tau_i}$ BuildTree$(S_i, h_i)$
4:     $\mathbf{W}^{(i)} \leftarrow$ WorkloadMatrix$(F_i; Q)$
5:     $\mathbf{A}^{(i)} \leftarrow \text{OPT}_{p_i}(\mathbf{W}^{(i)})$
6:     $\mathbf{B}^{(i)} \leftarrow \mathbf{W}^{(i)} \left(\mathbf{A}^{(i)}\right)^{+} \text{Lap}(q||\mathbf{A}^{(i)}||_1/\epsilon)^{m_i \times k}$
7:     $F_i \leftarrow$ UpdateLeaves$(X, F_i)$
8:     if DP Training then
9:         Let $\mathbf{Y}^{(i)}$ be leaf values of $F_i$
10:        $\tilde{F_i} \leftarrow$ update $F_i$ with leaf values $\mathbf{Y}^{(i)} + \mathbf{B}^{(i)}$
11:        $\tilde{\mathbf{V}}^{(i)} \leftarrow \tilde{F_i}(Q)$
12:    else if DP Prediction then
13:        $\mathbf{V}^{(i)} \leftarrow F_i(Q)$; $\tilde{\mathbf{V}}^{(i)} \leftarrow \mathbf{V}^{(i)} + \mathbf{B}^{(i)}$
14:    end if
15: end for
16: $\tilde{\mathbf{y}}_i = \arg\max_{1 \le l \le k} \sum_{j=1}^{q} \tilde{\mathbf{V}}^{(j)}_{i,l}$, $\forall$ inference query $i$
```

Setting Hyperparameters. Following Theorem 4 and 6, the utility of Algorithm 3 is measured by $\text{Err}_{\epsilon/q}(\mathbf{W}^{(i)}, \mathbf{A}^{(i)})$. This metric corresponds to errors in leaf values for DP training and vote counts for DP prediction. The error rates are affected by the matrix $\mathbf{W}^{(i)}$, which is defined by the number of trees τ_i, depth h_i, and schema information, including the number of features $\bar{d}$. Since this error metric does not depend on the actual training data, we can effectively perform *privacy-free hyperparameter tuning* without consuming any privacy budget.

The hyperparameters q (number of ensembles) and $\bar{d}$ (size of the feature subspace) involve a trade-off between accuracy and efficiency. Smaller feature sets enable more efficient optimizations but might exclude important features, reducing learning capability. Increasing the number of ensembles reduces the privacy budget per ensemble, potentially degrading accuracy. For smaller datasets, we recommend using a single ensemble forest ($q = 1$) with the original feature set ($\bar{d} = |S|$) to maximize accuracy. For larger datasets, $\bar{d}$ and q should be chosen to balance efficiency and accuracy.

Experiments

We empirically evaluate the performance of our DP training and prediction techniques for random forests, demonstrating higher accuracy compared to competing techniques.

Experimental Setup

Datasets. We use six popular classification datasets from the UCI ML Repository (Kelly, Longjohn, and Nottingham 2023) with feature dimensions ranging from 4 to 128: Car, Iris, Scale, Adult, Heart, and Mushroom. Certain datasets with continuous values were preprocessed using public domain knowledge, including discretization. The Adult dataset was already discretized (Chang and Lin 2011).

Implementation and Competing Techniques. We evaluate Algorithm 3 against various competing techniques. For consistency, all DP methods use random decision trees. All implementations are in Python and experiments were conducted on a MacBook Air (M2 chip with 16GB RAM). We adopt a multi-way tree structure as in ID3. It can be easily extended to binary trees.

We compare our DP batch training against two widely-used methods. The first baseline employs the same batch training but applies the Laplace mechanism, commonly used in state-of-the-art works (Jagannathan, Pillaipakkamnatt, and Wright 2009; Maddock et al. 2022). This corresponds to Algorithm 3 with $\mathbf{B}^{(i)} = \text{Lap}(q\tau_i/\epsilon)^{o \times k}$ and $q = 1, \bar{d} = |S|$. The second baseline trains each tree on disjoint datasets using the Laplace mechanism, as seen in prior works (Holohan et al. 2019; Fletcher and Islam 2017). For all methods above, we adopt the standard hard voting for prediction. We do not compare against work that uses a weaker definition of DP, such as (Rana, Gupta, and Venkatesh 2015). For DP prediction techniques, we compare our DP batch prediction and subsample-and-aggregate approach from Algorithm 3 against the existing subsample-and-aggregate (Dwork and Feldman 2018). Additionally, we benchmark our techniques against an optimized non-private random forest algorithm, the Extra-Trees classifier from scikit-learn to obtain an empirical upper bound on accuracy for private algorithms (Pedregosa et al. 2011). For runtime comparisons, we evaluate the non-private version of Algorithm 3 to ensure consistency.

Results

We measure the accuracy and runtime of Algorithm 3 with at least 5 trials under fixed parameters. Unless explicitly denoted, each dataset is split into train and test subsets with a 80:20 ratio. Each ensemble consists of τ/q trees of depth h.

Main Results. Figure 2 shows the test accuracy of our DP batch training and prediction techniques, varying values of ϵ. We consider commonly used values of $\tau = 64 \sim 128$. We use $q = 1, \bar{d} = d-1$ for small to medium-sized datasets like Car, Iris, and Balance; for larger datasets, multiple ensembles are employed. Our optimized approaches consistently outperform the baselines, showing a significant accuracy improvement of 20-40% for small to medium-sized datasets. Larger datasets such as Heart, Mushroom, and Adult show a notable accuracy improvement of 10-20%, particularly with smaller ϵ values. Our novel matrix representation improves DP batch training and prediction, enabling the learning of numerous base classifiers and the prediction of a large number of test samples, all without sacrificing accuracy.

Increasing the number of trees does not negatively impact the accuracy of DP Batch training. With random forests, predictive power is supposed to increase with the number of trees. However, DP batch and disjoint training with the Laplace mechanism both suffer from tree scaling issues, making the use of a smaller number of trees optimal. Batch training with the Laplace mechanism employs equal budget allocation; as the number of trees increases,

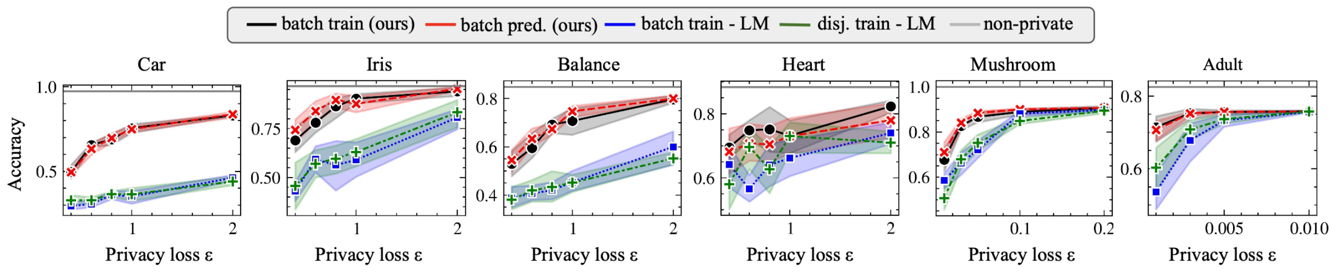


Figure 2: Test accuracy of different private prediction techniques on various datasets, varying values of privacy loss ϵ with fixed parameters:$h = 4, \tau = 128$ for Car, $h = 2, \tau = 64$ for Iris, $h = 2, \tau = 128$ for Balance, $h = 2, \tau = 128, q = 2, \bar{d} = 4$ for Heart, $h = 3, \tau = 125, q = 5, \bar{d} = 4$ for Mushroom and $h = 8, \tau = 100, q = 4, \bar{d} = 10$ for Adult.

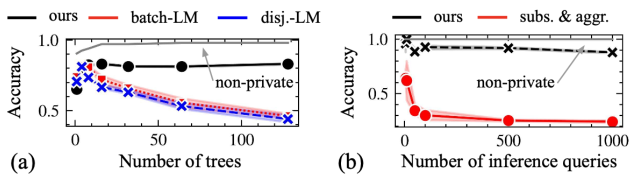


Figure 3: a) Test accuracy of DP training methods on the Car dataset vs. number of trees ($\epsilon = 2, h = 4, q = 1$). b) Accuracy of DP prediction methods on the Car dataset vs. number of inference queries ($\epsilon = 2, h = 4, \tau = 16, q = 1$).

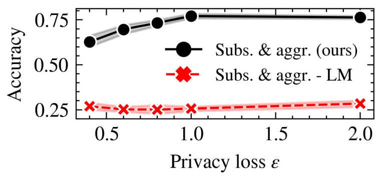


Figure 4: Test accuracy of different subsample-and-aggregate approaches on the Car dataset with various privacy loss budget of ϵ when $\tau = 16, h = 3, q = 1$.

Dataset	d	Method	Time (sec) Opti.	Train.	Pred.
Adult	124	DP Train.	255.7	12.84	30.09
		DP Pred.	58.58	12.88	29.63
		Non-DP	N/A	13.20	29.24
Heart	14	DP Train.	0.60	1.05	0.52
		DP Pred.	0.79	0.82	0.57
		Non-DP	N/A	0.72	0.36
Mushroom	23	DP Train.	16.24	12.22	10.17
		DP Pred.	24.68	11.97	10.07
		Non-DP	N/A	13.09	9.82

Table 1: Runtime of different privacy prediction techniques for datasets with feature dimension d, broken down into noise optimization, training, and prediction.

the privacy budget per tree decreases. The same tree scaling issue occurs during disjoint training; as the number of trees increases, the number of training samples per tree decreases. In both cases, the predictive power of a single tree is inversely proportional to the total number of trees. On the other hand, our DP batch training method preserves good accuracy as the number of trees increases, reflecting the scaling principle of ensemble learning; see Figure 3a.

Increasing the number of test samples has minimal impact on the accuracy of DP batch prediction. Figure3b illustrates the relationship between accuracy and the number of inference queries for the Car dataset, comparing our DP batch prediction against the baseline subsample-and-aggregate. Because of the limited number of samples, we use the entire Car dataset to train a model and report the prediction accuracy for 5 to 1000 inference queries. The existing approach exhibits low accuracy as the number of inference queries increases, reaching 30% accuracy for 100 samples. The per-sample budget allocation leads to an immediate degradation in accuracy. In contrast, with our DP batch prediction, accuracy remains at 90% even when predicting as many as 1000 samples. This effectively addresses potential challenges that existing DP prediction approaches face when making predictions on numerous samples.

Applying to the subsample-and-aggregate framework. We show the generalizability of our techniques by adapting them to the subsample-and-aggregate framework. Figure 4 shows the test accuracy of our optimized approach compared to the existing non-optimized version, when performing prediction on 345 examples with the Car dataset. Our technique improves the accuracy of existing solutions by up to 50%, with similar improvement in other datasets.

Runtime. Table 1 compares the runtime of Algorithm 3 to non-private random decision trees with the same hyperparameters as in Figure 2. The runtime for data-independent tree building is excluded as it incurs no DP overhead. The primary overhead comes from optimization, which is crucial for high accuracy. Despite this, our approach maintains minimal runtime overhead during training and prediction. Although our DP baselines have similar training and prediction complexities, they lack noise optimization. Our results show that the DP M-RF algorithm is feasible on commodity hardware, even with large datasets. The optimization overhead can be reduced by reducing the number of features per ensemble, potentially at the expense of some accuracy. Further experiments are detailed in the technical appendix.

Acknowledgments

This work was supported in part by the NSF grant 2131910.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 308–318.

Bassily, R.; Thakkar, O.; and Guha Thakurta, A. 2018. Model-agnostic private learning. *Advances in Neural Information Processing Systems*, 31.

Chang, C.-C.; and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2: 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3).

Dwork, C.; and Feldman, V. 2018. Privacy-preserving prediction. In *Conference On Learning Theory*, 1693–1702. PMLR.

Dwork, C.; and Lei, J. 2009. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 371–380.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407.

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

Fletcher, S.; and Islam, M. Z. 2017. Differentially private random decision forests using smooth sensitivity. *Expert systems with applications*, 78: 16–31.

Fletcher, S.; and Islam, M. Z. 2019. Decision tree classification with differential privacy: A survey. *ACM Computing Surveys (CSUR)*, 52(4): 1–33.

Geurts, P.; Ernst, D.; and Wehenkel, L. 2006. Extremely randomized trees. *Machine learning*, 63: 3–42.

Holohan, N.; Braghin, S.; Mac Aonghusa, P.; and Levacher, K. 2019. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints*, 1907.02444 [cs.CR].

Hou, J.; Li, Q.; Meng, S.; Ni, Z.; Chen, Y.; and Liu, Y. 2019. DPRF: a differential privacy protection random forest. *Ieee Access*, 7: 130707–130720.

Hu, H.; Salcic, Z.; Sun, L.; Dobbie, G.; Yu, P. S.; and Zhang, X. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s): 1–37. Publisher: ACM New York, NY.

Jagannathan, G.; Pillaipakkamnatt, K.; and Wright, R. N. 2009. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*, 114–121. IEEE.

Jayaraman, B.; and Evans, D. 2019. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, 1895–1912.

Kelly, M.; Longjohn, R.; and Nottingham, K. 2023. The UCI Machine Learning Repositor.

Li, C.; Miklau, G.; Hay, M.; McGregor, A.; and Rastogi, V. 2015. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24: 757–781.

Li, X.; Qin, B.; Luo, Y.; and Zheng, D. 2022. A Differential Privacy Budget Allocation Algorithm Based on Out-of-Bag Estimation in Random Forest. *Mathematics*, 10(22): 4338.

Maddock, S.; Cormode, G.; Wang, T.; Maple, C.; and Jha, S. 2022. Federated boosted decision trees with differential privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2249–2263.

McKenna, R.; Miklau, G.; Hay, M.; and Machanavajjhala, A. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10): 1206–1219.

Nissim, K.; Raskhodnikova, S.; and Smith, A. 2007. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 75–84.

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

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.; and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12: 2825–2830.

Ponomareva, N.; Hazimeh, H.; Kurakin, A.; Xu, Z.; Denison, C.; McMahan, H. B.; Vassilvitskii, S.; Chien, S.; and Thakurta, A. G. 2023. How to dp-fy ml: A practical guide to machine learning with differential privacy. *Journal of Artificial Intelligence Research*, 77: 1113–1201.

Rana, S.; Gupta, S. K.; and Venkatesh, S. 2015. Differentially private random forest with high utility. In *2015 IEEE international conference on data mining*, 955–960. IEEE.

Shokri, R.; Stronati, M.; Song, C.; and Shmatikov, V. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, 3–18. IEEE.

van der Maaten, L.; and Hannun, A. 2020. The trade-offs of private prediction. *arXiv preprint arXiv:2007.05089*.

Xiao, X.; Wang, G.; and Gehrke, J. 2010. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering*, 23(8): 1200–1214.

Xiao, Y.; Gardner, J.; and Xiong, L. 2012. Dpcube: Releasing differentially private data cubes for health information. In *2012 IEEE 28th International Conference on Data Engineering*, 1305–1308. IEEE.