Technical Appendix

Anonymous submission

Random Decision Trees Algorithm

Figure 1 shows the random decision trees algorithm which we utilize in our work. Random decision trees consist of τ trees with depth h. Each tree consists of internal and leaf nodes, representing tests on attributes and class labels, respectively. The construction of these random decision trees is similar to traditional decision trees, with the distinction of utilizing ID3-style random attribute selection at each internal node (refer to BuildTree in Algorithm 1). After the trees are built, class labels are assigned to every leaf node by fitting the data via UpdateLeaves in Algorithm 1. Each instance in X is tested along the root-to-leaf classification rules, and the most frequent class is assigned as a leaf label. While we consider multi-way tree structure as in ID3 (Quinlan 1986), it can be easily extended to binary trees in the case of CART (Breiman et al. 1984) or C4.5 (Quinlan 1993).

Sensitivity Analysis of Matrix Multiplication

In this section, we analyze the sensitivity of matrix multiplication in the context of random forest applications.

Vectorization of Matrix Operations

The core of our approach is to represent random forest training and prediction using the matrix multiplication form $\mathbf{Y} = \mathbf{W}\mathbf{X}_X$, where $\mathbf{W} \in \mathbb{R}^{l \times n}$ is a data-independent query matrix and $\mathbf{X}_X \in \mathbb{R}^{n \times k}$ is a matrix dependent on training data $X \in \mathcal{D}$. While we use the matrix form $\mathbf{X}$ for clarity, the vectorized form is often more convenient for sensitivity and/or privacy analysis. Specifically, we will often use the vectorized expression $\mathrm{vec}(\mathbf{Y}) = \mathrm{vec}(\mathbf{W}\mathbf{X}) = (\mathbf{I}_k \otimes \mathbf{W})\mathrm{vec}(\mathbf{X})$ when analyzing query sensitivity.

Throughout our work, $\mathrm{vec}(\mathbf{X})$ denotes the vectorization operation where the columns of the matrix are appended. $\mathbf{I}_k$ represents an identity matrix of size k and $\otimes$ denotes a kronecker product.

Sensitivity Analysis

We define the L1 sensitivity of $\mathbf{Y} = \mathbf{W}\mathbf{X}_X$ using its vectorized expression as follows.

Proposition 1. *Given l by n matrix $\mathbf{W}$ and n by k matrix $\mathbf{X}_X$ depending on data $X \in \mathcal{D}$ such that L1 sensitivity of $vec(\mathbf{X})$ is γ, let $f(X) = \mathrm{vec}(\mathbf{W}\mathbf{X}_X)$. The L1 sensitivity of f is $\gamma||\mathbf{W}||_1$.*

Algorithm 1: Random Decision Trees Classifier

function RandomForest(training data X, set of features S, depth h, number of trees τ)

for $i = 1, \ldots, \tau$ **do**
  $T_i \leftarrow$ BuildTree(S, h)
end for
$F \leftarrow$ UpdateLeaves$(X, \{T_1, \ldots, T_\tau\})$
Return F

function BUILDTREE(set of features S, depth h)

$T \leftarrow \{\}$
if $h > 0$ **then**
  uniformly randomly select a feature $A \in S$ to split the node.
  for $x \in \Phi(A)$ **do**
    $T \leftarrow T \cup$ BUILDTREE$(S - A, h - 1)$
  end for
end if
Return T

function UPDATELEAVES(training data X, forest F)

for every tree $T \in F$ **do**
  for every leaf i in T **do**
    $C_{i,j} \leftarrow$ count the number of instances labeled with class j in X that reach the leaf i for every $j \in \mathcal{Y}$.
    Compute a leaf label $L_i = \arg\max_{j \in \mathcal{Y}} C_{i,j}$.
  end for
end for
Return F

Proof. The L1 sensitivity of f is:

$$\begin{aligned}
\Delta f &= \max_{X \sim X' \in \mathcal{D}} ||f(X) - f(X')||_1 \\
&= \max_{X \sim X' \in \mathcal{D}} ||\mathrm{vec}(\mathbf{W}\mathbf{X}_X) - \mathrm{vec}(\mathbf{W}\mathbf{X}_{X'})||_1 \\
&= \max_{X \sim X' \in \mathcal{D}} ||(\mathbf{I}_k \otimes \mathbf{W})\mathrm{vec}(\mathbf{X}_X) - (\mathbf{I}_k \otimes \mathbf{W})\mathrm{vec}(\mathbf{X}_{X'})||_1 \\
&\leq ||\mathbf{I}_k \otimes \mathbf{W}||_1 \max_{X \sim X' \in \mathcal{D}} ||\mathrm{vec}(\mathbf{X}_X) - \mathrm{vec}(\mathbf{X}_{X'})||_1 \\
&\leq ||\mathbf{W}||_1 \gamma.
\end{aligned}$$

□

In the special case where $\mathbf{X} = \mathbf{D}$ (as defined in the main body of the paper), the sensitivity is $||\mathbf{W}||_1$ (since $\gamma = 1$).

More details on Matrix Mechanism

In this section, we provide detailed background on Matrix Mechanism, along with its privacy analysis.

Vectorized Laplace Mechanism. Consider the task of answering $\mathbf{Y} = \mathbf{WD}$ under DP. The standard approach involves employing the following Vectorized Laplace mechanism. There, $\text{Lap}(b)^{l\times k}$ denotes a matrix of lk independent samples from the Laplace distribution with scale b. The sensitivity of a set of queries defined by $\mathbf{W}$ is denoted as the L1 norm of $\mathbf{W}$, i.e., $||\mathbf{W}||_1$; see the above section as well.

Definition 2 (Vectorized Laplace Mechanism)**.** Given an l by n data-independent query matrix $\mathbf{W}$ and an n by k data matrix $\mathbf{D}$, the Laplace mechanism for answering $\mathbf{WD}$ satisfies ϵ-DP: $\text{LM}(\mathbf{W}, \mathbf{D}) = \mathbf{WD} + \text{Lap}(||\mathbf{W}||_1/\epsilon)^{m\times k}$.

Matrix Mechanism with Data Matrix. A more sophisticated approach involves identifying a set of strategy queries defined by the matrix $\mathbf{A}$ that supports the input query matrix $\mathbf{W}$. The Matrix Mechanism serves as a general mechanism for addressing such query matrices. By *selecting* a good strategy $\mathbf{A} \in \mathbb{R}^{m\times n}$, we *measure* it with the Laplace mechanism as $\tilde{\mathbf{Y}} = \mathbf{AD} + \text{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times n}$. We find the estimates $\tilde{\mathbf{D}}$ of the input data $\mathbf{D}$ by solving the least-squares problem of $\tilde{\mathbf{Y}} = \mathbf{AD}$. The final query answers can then be *reconstructed* as $\mathbf{W}\tilde{\mathbf{D}}$. Notably, this select-measure-reconstruct method is equivalently expressed as:

$$\tilde{\mathbf{Y}} := \mathbf{WD} + \mathbf{WA}^{+}\text{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}. \quad (1)$$

Above, the sensitivity of a query matrix is defined as its L1 norm denoted as $||\mathbf{A}||_1$. In the original Matrix Mechanism paper (Li et al. 2015), a data vector was utilized instead of a data matrix, although both expressions are essentially interchangeable, as we discussed in the previous section. Namely, for matrices $\mathbf{D} \in \mathbb{R}^{n\times k}, \mathbf{W} \in \mathbb{R}^{l\times n}, \mathbf{A} \in \mathbb{R}^{m\times n}$ in the above expression, consider substituting with $\mathbf{x} = \text{vec}(\mathbf{D}) \in \mathbb{R}^{nk}$, $\mathbf{W}_\otimes = \mathbf{I}_k \otimes \mathbf{W} \in \mathbb{R}^{lk\times nk}$, $\mathbf{A}_\otimes = \mathbf{I}_k \otimes \mathbf{A} \in \mathbb{R}^{mk\times nk}$ respectively, resulting in the following expression with a data vector:

$$\tilde{\mathbf{y}} := \mathbf{W}_\otimes\mathbf{x} + \mathbf{W}_\otimes\mathbf{A}_\otimes^{+}\text{Lap}(||\mathbf{A}_\otimes||_1/\epsilon)^{mk}. \quad (2)$$

Proposition 3. *The vectorization of Equation 1 is equivalent to Equation 2.*

Proof. We show that $\tilde{\mathbf{y}} = \text{vec}(\tilde{\mathbf{Y}})$, where $\tilde{\mathbf{y}}$ is from Equation 2 and $\tilde{\mathbf{Y}}$ is from Equation 1:

$$\begin{aligned}
\tilde{\mathbf{y}} &= \mathbf{W}_\otimes\mathbf{x} + \mathbf{W}_\otimes\mathbf{A}_\otimes^{+}\text{Lap}\left(||\mathbf{A}_\otimes||_1/\epsilon\right)^{mk} \\
&= \left(\mathbf{I}_k \otimes \mathbf{W}\right)\text{vec}\left(\mathbf{D}\right) \\
&+ \left(\mathbf{I}_k \otimes \mathbf{W}\right)\left(\mathbf{I}_k \otimes \mathbf{A}\right)^{+}\text{Lap}\left(||\left(\mathbf{I}_k \otimes \mathbf{A}\right)||_1/\epsilon\right)^{mk} \\
&= \text{vec}\left(\mathbf{WD}\right) + \left(\mathbf{I}_k \otimes \mathbf{WA}^{+}\right)\text{Lap}(||\mathbf{A}||_1/\epsilon)^{mk} \\
&= \text{vec}\left(\mathbf{WD}\right) + \left(\mathbf{I}_k \otimes \mathbf{WA}^{+}\right)\text{vec}\left(\text{Lap}\left(||\mathbf{A}||_1/\epsilon\right)^{m\times k}\right) \\
&= \text{vec}\left(\mathbf{WD}\right) + \text{vec}\left(\mathbf{WA}^{+}\text{Lap}\left(||\mathbf{A}||_1/\epsilon\right)^{m\times k}\right) \\
&= \text{vec}\left(\mathbf{WD} + \mathbf{WA}^{+}\text{Lap}\left(||\mathbf{A}||_1/\epsilon\right)^{m\times k}\right) \\
&= \text{vec}\left(\tilde{\mathbf{Y}}\right)
\end{aligned}$$

□

Privacy Analysis. The privacy of our DP random forest training and prediction primarily follows from the privacy of Matrix Mechanism. In the select-measure-reconstruct approach, the data-independent selection step consumes zero privacy budget, the measurement step satisfies ϵ-DP, following from the privacy of the Laplace mechanism, and the reconstruction step does not compromise privacy, due to the post-processing theorem. This analysis forms the basis for proving the privacy claims in Theorem 6 and 9.

More Details on Our Optimized Subsample-and-Aggregate Framework

In this section, we provide details of our enhanced subsample-and-aggregate framework for random decision trees using our matrix representation. In addition to the weight voting mechanism introduced in the main paper, we consider the standard hard voting mechanism.

Random Decision Trees with Disjoint Datasets The subsample-and-aggregate framework is a DP prediction technique in which DP noise is added to aggregated votes during prediction (Dwork and Feldman 2018). The primary distinction between this approach and our DP batch prediction presented is that the subsample-and-aggregate framework uses a disjoint subset of the training data to train each tree, while the DP batch prediction utilizes the full dataset.

Consider τ random decision trees with training data X. The training data is randomly split into τ disjoint subsets $X_1, \ldots, X_\tau$, where X_i is utilized to train the i-th tree. For prediction on a test sample, we aggregate predicted output labels from the ensemble of trees, considering both weight voting and the standard hard voting mechanisms. Below, we present our approaches to performing weight voting and hard voting under DP through matrix operations.

DP Weight Voting Approach

In this section, we show our subsample-and-aggregate framework for performing weight voting under DP using matrices. Our goal is to estimate weighted votes $\mathbf{V} = \mathbf{QT}^\intercal\mathbf{C}$ under DP while maintaining good accuracy. Recall that $\mathbf{Q}, \mathbf{T}^\intercal, \mathbf{C}$ represent inference query matrix, decision path matrix, and leaf value matrix.

Suboptimal Sensitivity Analysis. We begin by considering the existing approach, which introduces Laplace noise to vector counts $\mathbf{V}_i$ for each inference query i with an equal budget allocation: each inference query receives a privacy budget of ϵ/b. The sensitivity of a single inference query is one, as the trees are trained on disjoint datasets, and the vector counts $\mathbf{V}_i$ can change by at most one at a single class location. Importantly, the tree structure is data-independent.

Definition 4 (Laplace Mechanism for Vectorized Weight Voting)**.** Given inference query matrix $\mathbf{Q} \in \mathbb{R}^{b\times n}$, decision path matrix $\mathbf{T} \in \mathbb{R}^{o\times n}$ and leaf value matrix $\mathbf{C} \in \mathbb{R}^{o\times k}$, the Laplace mechanism for answering vote counts for each class label, satisfying ϵ-DP, is defined as follows: $\tilde{\mathbf{V}} = \mathbf{QT}^\intercal\mathbf{C}_\mathbf{D} + \text{Lap}(b/\epsilon)^{b\times k}$.

However, this sensitivity analysis on a per-query basis may be suboptimal. Instead, we perform a sensitivity analysis on a batch of queries represented by $\mathbf{QT}^\intercal$, whose sensitivity can be computed as the L1 norm of the matrix. For instance, when two inference queries do not share the same decision paths in the random forest, the sensitivity can be analyzed as 1 instead of 2, as data points used for voting do not overlap between the two queries.

DP Weight Voting with Strategy Matrix. Our matrix representation enables an accurate computation of query sensitivity. We introduce an enhanced approach for measuring vote counts under DP by identifying an optimal strategy matrix $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{QT}^\intercal)$ that minimizes the error of $\mathbf{QT}^\intercal\mathbf{C}$, i.e., $\mathrm{Err}(\mathbf{QT}^\intercal, \mathbf{A})$ (the error term is defined in the main paper). The resulting DP answers to $\mathbf{QT}^\intercal$ on the leaf value matrix $\mathbf{C}$ are computed as follows.

Definition 5 (DP Weight Voting via Strategic Decision Paths)**.** Given inference query matrix $\mathbf{Q} \in \mathbb{R}^{b\times n}$, decision path matrix $\mathbf{T} \in \mathbb{R}^{o\times n}$ and leaf value matrix $\mathbf{C_D} \in \mathbb{R}^{o\times k}$, and privacy budget ϵ, let $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{QT}^\intercal) \in \mathbb{R}^{m\times n}$. DP vote counts for the inference queries are answered by:

$$\tilde{\mathbf{V}} = \mathbf{QT}^\intercal\mathbf{C_D} + \mathbf{QT}^\intercal\mathbf{A}^{+}\mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}.$$

Theorem 6. *Definition 5 satisfies ϵ-DP.*

Proof. The privacy analysis primarily follows from the privacy of Matrix Mechanism. All we need to show is that the measurement step $\mathbf{AC} + \mathrm{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$ satisfies ϵ-DP. In the subsample-and-aggregate framework, adding or removing an individual sample can only change leaf counts defined by $vec(\mathbf{C})$ by at most one at one location, and thus can change the weight counts defined by $\mathrm{vec}(\mathbf{AC})$ at most $||\mathbf{A}||_1$, following Theorem 1. Here, we consider the vectorized form for the convenience of privacy analysis. Thus, from the privacy of Laplace mechanism, the measurement step satisfies ϵ-DP. Therefore, from the privacy analysis of Matrix Mechanism, Definition 5 satisfies ϵ-DP. □

DP Hard Voting Approach

In this section, we present our subsample-and-aggregate framework for performing hard voting under DP using matrix representation.

Hard Voting Matrix Operation. In hard voting, each tree casts a unit vote for a predicted class label, and these votes are aggregated across all trees. Similar to weight voting, we can represent hard voting using matrix multiplication. We define leaf labels as a matrix $\mathbf{L} \in \mathbb{R}^{o\times k}$ using one-hot encoding: $\mathbf{L}_{i,j} = 1$ if the output label at the i-th leaf node is $y_j \in \mathcal{Y}$, and 0 otherwise. This matrix may be explicitly denoted as $\mathbf{L_D}$ to indicate its dependence on the training data $\mathbf{D}$. Utilizing the inference query matrix $\mathbf{Q}$, decision path matrix $\mathbf{T}$, and leaf label matrix $\mathbf{L_D}$, the aggregation of vote counts across trees for every inference query can be expressed as the following matrix multiplication: $\mathbf{V} = \mathbf{QT}^\intercal\mathbf{LD} \in \mathbb{R}^{b\times k}$.

Suboptimal Sensitivity Analysis. We consider the following baseline mechanism for hard voting in the subsample-and-aggregate framework. This method adds Laplace noise to the vector counts $\mathbf{V}_i$ for each inference query, with an equal budget allocation of ϵ/b for each query. Each query can change the vector counts $\mathbf{V}_i$ by at most one at *two* class label locations, resulting in a sensitivity of two. Note that since trees are trained on disjoint datasets, the addition or removal of one record can only affect a single tree.

Definition 7 (Laplace Mechanism for Vectorized Hard Voting)**.** Given inference query matrix $\mathbf{Q} \in \mathbb{R}^{b\times n}$, decision path matrix $\mathbf{T} \in \mathbb{R}^{o\times n}$ and leaf label matrix $\mathbf{L} \in \mathbb{R}^{o\times k}$, the following Laplace mechanism for answering vote counts for each class label satisfies ϵ-DP: $\tilde{\mathbf{V}} = \mathbf{QT}^\intercal\mathbf{L_D} + \mathrm{Lap}(2b/\epsilon)^{b\times k}$.

DP Hard Voting via Strategy Matrix. We refine the sensitivity analysis of hard voting for the subsample-and-aggregate framework in a similar fashion to weight voting. The enhanced strategy for measuring votes under DP is presented as follows.

Definition 8 (DP Hard Voting via Strategic Decision Paths)**.** Given inference query matrix $\mathbf{Q} \in \mathbb{R}^{b\times n}$, decision path matrix $\mathbf{T} \in \mathbb{R}^{o\times n}$ and leaf label matrix $\mathbf{L_D} \in \mathbb{R}^{b\times k}$, and privacy budget ϵ, let $\mathbf{A} \leftarrow \mathrm{OPT}(\mathbf{QT}^\intercal) \in \mathbb{R}^{m\times n}$. DP vote counts for the inference queries are answered by:

$$\tilde{\mathbf{V}} = \mathbf{QT}^\intercal\mathbf{L_D} + \mathbf{QT}^\intercal\mathbf{A}^{+}\mathrm{Lap}(2||\mathbf{A}||_1/\epsilon)^{m\times k}.$$

Theorem 9. *Definition 8 satisfies ϵ-DP.*

Proof. The privacy analysis primarily follows from the privacy of the Matrix Mechanism. All we need to show is that the measurement step $\mathbf{AL} + \mathrm{Lap}(2||\mathbf{A}||_1/\epsilon)^{m\times k}$ satisfies ϵ-DP. In the subsample-and-aggregate framework, adding or removing an individual sample can only change $\mathrm{vec}(\mathbf{L})$ by at most one at *two* locations, and thus can change the values defined by $\mathrm{vec}(\mathbf{AL})$ at most $2||\mathbf{A}||_1$, following Theorem 1. Here, for the convenience of privacy analysis, we consider the vectorized form. With this sensitivity analysis and the privacy of Laplace mechanism, the above measurement step satisfies ϵ-DP. Therefore, from the privacy analysis of Matrix Mechanism, it is evident that Definition 5 satisfies ϵ-DP. □

Subsample-and-Aggregate Framework for M-RF

We show our improved subsample-and-aggregate framework for DP random forest prediction in Algorithm 2. Non-private random decision trees are trained with disjoint subsets of training data via Algorithm 1. The resulting tree structure and leaf values are transformed into matrices $\mathbf{T}, \mathbf{C}$. Similarly, b inference queries are represented as matrix $\mathbf{Q}$. Using these matrices, we estimate votes $\mathbf{V}$ under DP via Definition 5 for weight voting and Definition 8 for hard voting. Finally, we assign class labels based on the DP vote counts and return those as prediction results.

Theorem 10. *Algorithm 2 satisfies ϵ-DP.*

Proof. Following Theorem 5 for weight voting (Theorem 8 for hard voting), vote counts $\tilde{\mathbf{V}}$ satisfy ϵ-DP. In addition, resulting labels obtained from the ϵ-DP votes satisfy ϵ-DP due

to the post-processing theorem. Thus, Algorithm 2 satisfies ϵ-DP. □

Algorithm 2: M-RF with the Subsample-and-Aggregate

Require: Leaf value matrix $\mathbf{C}$, decision path matrix $\mathbf{T}$, inference query matrix $\mathbf{Q}$, privacy budget ϵ, voting mechanism vote $\in \{\text{weight}, \text{hard}\}$.

Ensure: DP predicted label vector $\tilde{\mathbf{y}}$

$\mathbf{A} \leftarrow \text{OPT}(\mathbf{QT}^\intercal)$
if vote $=$ weight **then**
  $\tilde{\mathbf{V}} = \mathbf{QT}^\intercal\mathbf{C} + \mathbf{QT}^\intercal\mathbf{A}^+\text{Lap}(||\mathbf{A}||_1/\epsilon)^{m\times k}$
end if
if vote $=$ hard **then**
  for every leaf i **do**
    $\mathbf{L}_{i,j} := \mathbb{I}(j = \arg\max_{1\le j\le k} \mathbf{C}_{i,j}), \forall j = 1, \ldots, k$ (One-hot vector; $\mathbb{I}$ is an identity function)
  end for
  $\tilde{\mathbf{V}} = \mathbf{QT}^\intercal\mathbf{L} + \mathbf{QT}^\intercal\mathbf{A}^+\text{Lap}(2||\mathbf{A}||_1/\epsilon)^{m\times k}$
end if
for every test sample i **do**
  $\tilde{\mathbf{y}}_i = \arg\max_{1\le j\le k} \tilde{\mathbf{V}}_{i,j}$
end for

Table 1: Datasets

Data	Records (N)	Features ($d-1$)	Classes (k)
Car Evaluation	1728	6	4
Iris	150	4	3
Balance Scale	625	4	3
Adult	30956	124	2
Heart Disease	297	13	2
Mushroom	8124	22	2

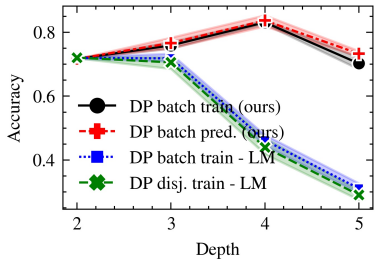


Figure 1: Test accuracy of various private prediction techniques on the Car dataset as a function of maximum depth with fixed $\epsilon = 2, \tau = 128$.

More Experiments

In this section, we provide details of our experimental setups and comprehensive empirical evaluation of our proposed algorithms, including the improved subsample-and-aggregate framework.

Datasets and Preprocessing

Figure 1 summarizes the key characteristics of the classification datasets used in our experiments, including the number of records, features, and classes for each dataset. During preprocessing, some datasets were processed using methods like discretization. In practice, such preprocessing must adhere to DP, either by utilizing a privacy budget or leveraging public domain knowledge. In our experiments, we assume the availability of domain knowledge, as is commonly done. For datasets with numerical features like Iris, and Heart, we applied equal-width binning to discretize the data.

In our experiments, the Car, Iris, and Balance datasets are considered small in terms of the number of features. Therefore, when evaluating our algorithms, we build a single ensemble forest over the entire feature set, i.e., $q = 1, \bar{d} = d - 1$. On the other hand, for the Adult, Heart, and Mushroom datasets, we employ multiple ensemble forests over random feature subsets, with the parameters q and $\bar{d}$ carefully chosen via our privacy-free hyperparameter search to balance efficiency and accuracy.

Implementation Details

We present implementation details along with optimizations for enhanced efficiency. Our implementation strategy involves recursively building trees and transforming relevant information, including the tree structure, into matrix forms to perform our optimization approaches. The executions of trees and optimization are parallelized across multiple ensembles with the maximum of 8 threads to boost speed. Memory efficiency is improved by optimizing our matrix representation. For instance, in our DP batch training, we compute $\mathbf{TD}$ without relying on matrix multiplication. Instead, we utilize the tree-building process through a recursive function, eliminating the need to explicitly represent the data matrix $\mathbf{D}$. While the query matrix $\mathbf{T}$ is crucial for optimization, we adopt an efficient representation method from (McKenna et al. 2018) to reduce the size from $o\Pi_{i=1}^{\bar{d}} n_i$ to $o\sum_{i=1}^{\bar{d}} n_i$, where n_i is the domain size of the i-th feature in a random feature subset of size $\bar{d}$; $\mathbf{o}$ denotes the number of leaf nodes in the entire ensemble forest.

Accuracy vs. Maximum Depth

Figure 1 shows test accuracy of our DP batch training and prediction techniques, varying the maximum depth h from 2 to 5 when $\epsilon = 2.0, \tau = 128$ for the Car dataset. We compare the performance against the baseline DP training techniques. While our optimized approaches show the best accuracy of 84% at depth 4, at the same depth parameter, the existing approaches show accuracy of 48%. The existing approaches has the optimal accuracy of 72% at depth of 2 and as the tree becomes deeper the accuracy degrades. Nevertheless, our techniques consistently outperform the baseline approach, regardless of tree depth.

Runtime

Figure 2 shows the runtime comparison of our DP batch training and prediction methods against non-DP random decision trees across various datasets. The total runtime for private prediction is broken down into tree building (Tree Build.), DP-noise optimization (OPT), training (Fitting), and

prediction (Predict), with DP-noise optimization not applicable to the non-DP baseline. Both tree building and noise optimization can be preprocessed without needing access to the training data.

The plots show that the primary overhead in our methods stems from noise optimization, which is essential for achieving accuracy. Despite this, our approach incurs minimal runtime overhead during the actual training and prediction phases, even with the added complexity of our matrix representation. The optimization cost depends on various factors such as random feature size (i.e.,$\bar{d}$), domain sizes, tree depths, and the number of inference queries, as detailed in the complexity analysis appeared in the main paper. For smaller datasets, the time spent on noise optimization is less than that for building the random decision trees.

Multiple ensembles. The large optimization cost seen with the Car dataset is due to building a single ensemble forest over the entire feature set. This inefficiency can be mitigated by using multiple ensembles, where each ensemble forest is built over a smaller subset of features, allowing for more efficient local optimization with a compact matrix. Optimizations across ensembles are in fact parallelizable. Table 2 summarizes the runtime of our DP random decision trees algorithm on the Car dataset with different hyperparameters, using a fixed value of $\epsilon = 1$. In the table, "Non-DP" refers to non-DP random decision trees. For each DP method, we fixed the total number of tree estimators at 128 but varied the number of ensembles: one with a single ensemble forest built over all six features, and another with two ensemble forests, each built over random subsets of four features.

Using multiple ensembles with random feature subsets significantly reduced optimization costs. Generally, a larger number of ensembles improves efficiency, although it may lead to accuracy drop. Interestingly, in the case of the Car dataset, using two ensembles actually improved accuracy over a single ensemble forest, underscoring the importance of selecting good hyperparameters, which can be done through our data-independent hyperparameter tuning.

Performance of our Optimized Subsample-and-Aggregate Framework

We evaluate Algorithm 2, compared against the existing subsample-and-aggregate framework (cf. Definition 4 and 7). We consider weight voting and hard voting. Figure 3 shows test accuracy of various datasets, varying values of ϵ with fixed depth h and number of trees τ: $h = 3, \tau = 16$ for Car, $h = 2, \tau = 16$ for Iris, and $h = 2, \tau = 16$ for Balance. Our optimized strategy consistently outperforms the state-of-the-art strategy, showing that the existing per-inference-query sensitivity analysis results in suboptimal accuracy. For instance, when $\epsilon = 1$, the Car dataset with about 340 test samples shows 75% accuracy for our optimized approach with weight voting and 25% accuracy for the existing method (50% improvement). The accuracy improvements against the baselines were similarly observed with different hyperparameters including tree depth and number of trees.

Accuracy vs. number of inference queries. The high accuracy of our approaches persists even with a larger number of inference queries, shown in Figure 4. There, we trained a model using the entire Car dataset and reported the training accuracy on various sizes of samples ranging from 5 to 1000. Fixed parameters, including $\epsilon = 2.0$, $h = 4$, and $\tau = 16$, were utilized. As shown in Figure 4, the baselines, which employ the per-sample budget allocation, lead to an immediate degradation in accuracy: predicting 50 samples results in 30-35% accuracy. In contrast, our optimized budget allocation maintains 70% for hard voting and 80% for weight voting even with a large number of samples such as 1000.

Weight voting vs. hard voting. When comparing among different voting systems, we observe that weight voting generally outperforms hard voting. However, the accuracy performance can significantly vary depending on chosen parameters. Figure 5 shows the test accuracy of the Car dataset when varying the tree depth and number of trees with $\epsilon = 2$ for our optimized subsample-and-aggregate framework. While weight voting tends to exhibit poor accuracy with deeper trees and larger number of trees, hard voting does not. Similar observations were made when the same experiment was performed on other datasets.

Comparison with DP batch training and prediction. The subsample-and-aggregate approach generally yields lower accuracy than our DP batch training and prediction techniques for small to medium-sized datasets, as shown in Table 2 for the Car dataset. This is due to the smaller number of tree estimators required in this approach; with training data partitioned across many trees, each tree gets fewer examples, limiting its learning capacity. However, for datasets with a large number of records, such as Adult, the subsample-and-aggregate method can achieve accuracy comparable to that of DP batch training and prediction.

References

Breiman, L.; Friedman, J.; Stone, J. C.; and Olshen, A. R. 1984. *Classification and Regression Trees*. Chapman and Hall/CRC.

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

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.

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.

Quinlan, J. R. 1986. Induction of decision trees. *Machine learning*, 1: 81–106.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN 1558602402.

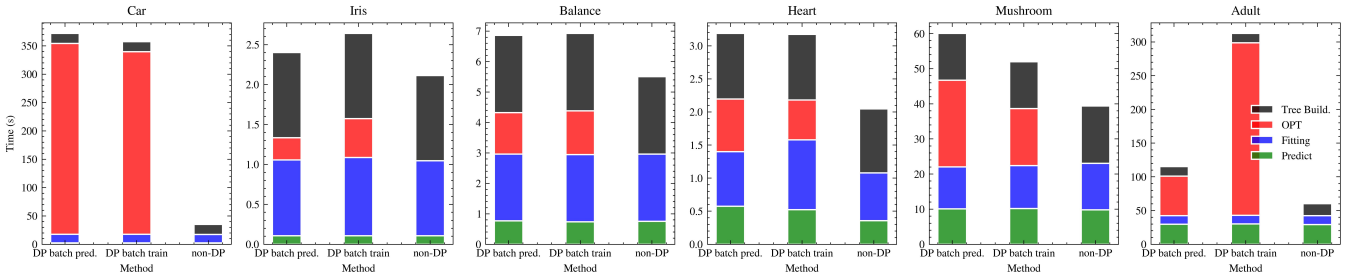


Figure 2: Runtime breakdown of our DP batch training and prediction methods compared to non-DP random decision trees using the same hyperparameters as in the main paper.

Table 2: Runtime and accuracy of our different DP methods with various hyperparameters on the Car dataset when $\epsilon = 1.0$. The hyperparameters include the number of ensembles q, feature subset size $\bar{d}$, number of trees τ and tree depth h.

Method	Hyperparameters				Time (sec)				Accuracy
	q	$\bar{d}$	τ	h	Tree Building	Noise optimization	Training	Prediction	
Non-DP	1	6	128	4	17.61	N/A	14.97	2.42	0.789
DP Batch Training	1	6	128	4	17.61	322.2	15.12	2.40	0.735
	2	4	128	4	17.07	84.97	14.96	2.54	0.810
DP Batch Prediction	1	6	128	4	17.61	336.6	15.20	2.53	0.793
	2	4	128	4	17.07	19.93	14.83	2.63	0.809
Subsample-and-aggregate	1	6	16	3	0.71	0.44	0.64	0.34	0.749
	2	4	16	3	0.84	0.66	0.72	0.50	0.763

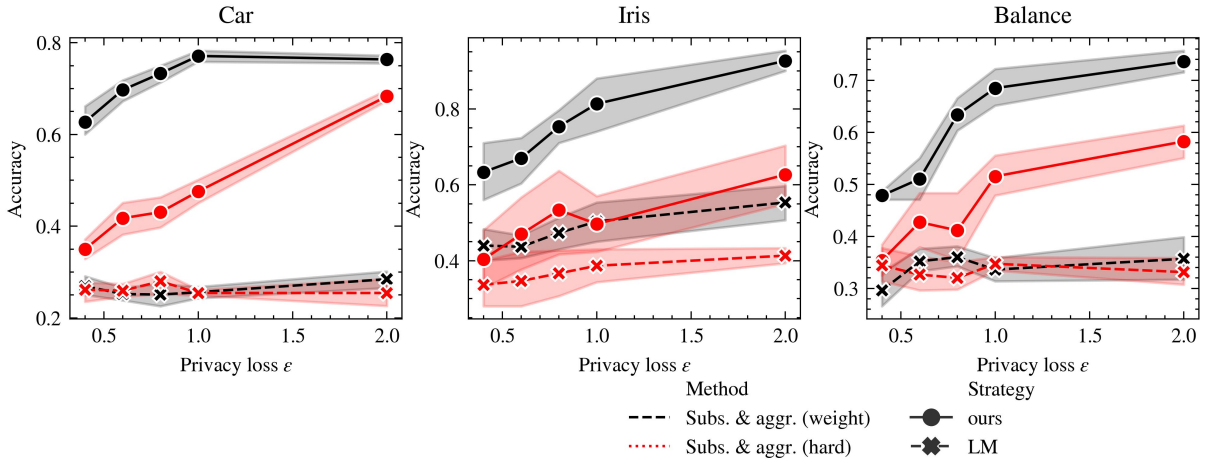


Figure 3: Test accuracy of different subsample-and-aggregate frameworks with weight and hard voting mechanisms on various datasets, varying values of ϵ with fixed depth h and number of trees τ: $h = 3, \tau = 16$ for Car, $h = 2, \tau = 16$ for Iris, and $h = 2, \tau = 16$ for Balance.

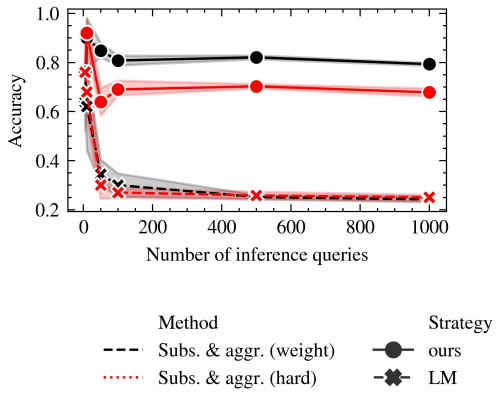
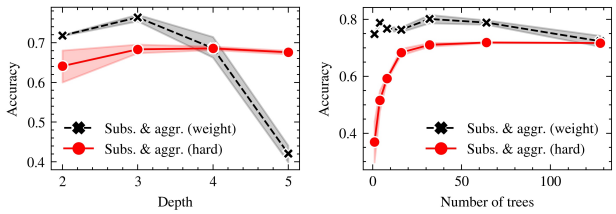


Figure 4: Training accuracy of different subsample-and-aggregate frameworks with weight and hard voting mechanisms on the Car dataset as a function of number of inference queries with $\epsilon = 2, h = 4, \tau = 16$.



(a) (b)

Figure 5: Test accuracy of different voting mechanisms of our optimized subsample-and-aggregate framework on the Car dataset as functions of depth and number of trees. Fixed parameters are: a) $\epsilon = 2.0, \tau = 16$ and b) $\epsilon = 2.0, h = 3$.