Privacy-Preserving Decision Trees Training and Prediction[⋆]

Adi Akavia[1][0000-0003-0853-3576], Max Leibovich[1], Yehezkel S. Resheff[2], Roey Ron[2], Moni Shahar[3], and Margarita Vald[2,4]

[1] University of Haifa, Israel {adi.akavia, max.fhe.phd}@gmail.com
[2] Intuit Inc., Israel {hezi.resheff, roey1rg}@gmail.com
[3] Facebook Inc., Israel monishahar@gmail.com
[4] Tel Aviv University, Israel margarita.vald@cs.tau.ac.il

Abstract. In the era of cloud computing and machine learning, data has become a highly valuable resource. Recent history has shown that the benefits brought forth by this data driven culture come at a cost of potential data leakage. Such breaches have a devastating impact on individuals and industry, and lead the community to seek privacy preserving solutions. A promising approach is to utilize Fully Homomorphic Encryption (FHE) to enable machine learning over encrypted data, thus providing resiliency against information leakage. However, computing over encrypted data incurs a high computational overhead, thus requiring the redesign of algorithms, in an "FHE-friendly" manner, to maintain their practicality. In this work we focus on the ever-popular tree based methods (e.g., boosting, random forests), and propose a new privacy-preserving solution to training and prediction for trees. Our solution employs a low-degree approximation for the step-function together with a lightweight interactive protocol, to replace components of the vanilla algorithm that are costly over encrypted data. Our protocols for decision trees achieve practical usability demonstrated on standard UCI datasets encrypted with fully homomorphic encryption. In addition, the communication complexity of our protocols is independent of the tree size and dataset size in prediction and training, respectively, which significantly improves on prior works.

Keywords: fully homomorphic encryption · privacy preserving machine learning · decision trees · training · prediction

1 Introduction

The ubiquity of data collected by products and services is often regarded as the key to the so called *AI revolution*. User and usage information

[⋆] An extended abstract for this work appeared in The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), September 14-18, 2020.

is aggregated across individuals to drive smart products, personalized experience, and automation. In order to achieve these goals, stored data is accessed by multiple microservices each performing different calculations. However, these benefits come at a cost of a threat on privacy.

The public is constantly being informed of data breaches, events which impact privacy and safety of individuals and in turn have a large negative effect on the breached service providers. Whether the leakage is passwords, private pictures and messages, or financial information, it is becoming increasingly clear that drastic measures must be taken to safeguard data that is entrusted to corporations.

There are several approaches to safeguarding data and minimizing the impact of potential breaches. Most fundamentally, encryption of data at rest ensures that even if the entire database is stolen, the data is still safe. While this may have sufficed in the past, the rise of microservice-based architectures in the cloud resulted in a large number of applications having access to the cleartext (i.e., unencrypted) information, making the attack surface uncontrollably large. Ideally, we would like to allow all these applications to operate without ever being exposed to the actual information. Recent advances in the field of Homomorphic Encryption provide some hope of achieving this level of privacy.

Fully Homomorphic Encryption (FHE) [33,16,5,4,15,9] is a type of encryption that allows computation to be performed over encrypted data ("homomorphic computation"), producing an encrypted version of the result. Concretely, FHE supports addition and multiplication over encrypted data, and hence allows evaluating any polynomial. The downside of FHE is the heavy cost of the multiplication operation, which imposes computational limitations on the degree of the evaluated polynomial and the number of total multiplications.

Unfortunately, common computations are not "FHE-friendly" as their polynomial representation is of high degree, which is a major obstacle to the widespread deployment of computing over encrypted data in practice. In particular, machine learning models require complex calculations to train and predict, and adaptations must be made in order to make them practical with FHE. Previous work on machine learning with FHE focused mostly on training and evaluation of logistic regression models, e.g., [7,22], and on more complex models such as shallow neural networks e.g., [17,29]. While these are two widely used classes of models, they are far from encompassing the entire scope of broadly used machine learning methods. In practice, tree based models remain some of the most popular methods, ranging from single decision trees, to random forests and boosting.

A decision tree is a model used for prediction, *i.e.*, mapping a feature vector to a score or a label. The prediction is done by traversing a path from root to leaf, where the path is determined by a sequence of comparison operations "$x_i > \theta$" between a feature value x_i and a threshold θ (continuing to right-child if satisfied, left otherwise). Training is the process of producing a decision tree from a dataset of labeled examples, with the goal of yielding accurate predictions on new unlabeled data instances.

Any solution for decision trees over encrypted data would need to address how to perform the comparison operations over such data. For prediction over encrypted data, Bost *et al.*[3] instantiated the comparison component via an interactive protocol, yielding communication complexity proportional to the tree size; subsequent work [6,1,41,10,37,20,24,38], likewise, followed the interactive approach with communication complexity proportional to the tree size or depth, imposing a significant burden on the bandwidth. In the context of training, existing protocols [27,12,40,42] [14,35,39,11,28] consider a multi-party setting where each party holds a cleartext subset of the dataset to be trained on, in contrast to our setting where the data is encrypted and no entity in the system holds in it the clear. This leaves the question of training decision trees over encrypted data together with non-interactive prediction as an open problem.

Elaborating on the above, this work is motivated by the enterprise setting, with a primary goal of providing a privacy-preserving solution compatible with the existing enterprise architecture. In this architecture, data is stored encrypted in a centralized storage, called *data lake*, and used by multiple microservices (referred to as server) that perform computations on cleartext data decrypted with a key provided by the enterprise key-management service (KMS). The KMS is an entity holding enterprise secrets and keys and providing crypto-services to authorized entities, and thus must be safeguarded. As part of its safeguarding, the KMS is restricted to a lightweight and predefined functionality, in particular, it is prohibited from executing heavy or general purpose code. Our goal is to completely eliminate the microservices access to cleartext data, and replace it with computation over encrypted data producing an encrypted outcome (that may either be decrypted by the KMS or used in encrypted form for subsequent computations). The KMS may be employed for computation on cleartext data, provided it adheres to the aforementioned restrictions on the KMS, in particular, in must be lightweight.

Our Contribution. In this work we present the first protocols for privacy-preserving decision tree based training and prediction that attain all the following desirable properties (see Figure 7-8 and Table 1 in Section 5):

1. *Prediction:* a non-interactive protocol on encrypted data.
2. *Training:* a d-round protocol between a server computing on encrypted data and the KMS, with communication complexity independent of the dataset size, where d is the constructed tree depth.
3. *Security:* provable privacy guarantees against an adversary who follows the protocol specification but may try to learn more information (semi-honest).
4. *Practical usability:* high accuracy comparable to the classical vanilla decision tree, fast prediction (seconds) and practical training (minutes to hours) demonstrated on standard UCI datasets encrypted with FHE.

Our technique for comparison over encrypted data. We devise a low degree polynomial approximation for step functions by using the least squares method, and utilize our approximation for fast and accurate prediction and training over encrypted data. To achieve better accuracy in our algorithms and protocols, the approximation uses a weighting function that is zero in a window around the step and constant elsewhere. See Section 3.1.

Further applications. Our training and prediction protocols can be employed in additional settings:
(a) Cross-entity: Our prediction protocol can trivially be used in settings where one company holds an unlabeled example, with the goal of learning the prediction result, and the other company holds a decision tree.
(b) Secure outsourcing: Both our protocols can be employed in settings where the client is the owner of example and tree in prediction (respectively, the dataset in training), and the server performs all computation (besides the lightweight KMS tasks performed by the client), resulting in protocols with lightweight client.

Terminology. Henceforth we use the more neutral "*client*" terminology rather than "KMS", in order to capture the aforementioned wider scope of applications.

Prior work on privacy-preserving decision trees. For *prediction*, prior works considered the cross-entity setting, presenting interactive protocols with communication complexity proportional to the tree size in [6,1,3,41,10,37,20,24] or depth [38]. In contrast, our protocol is non-interactive.

For *training*, the prior works [27,12,40,42,14,35,39,11,28] considered multi-party computation settings, where multiple parties communicate to train a model on the union of their private individual datasets with the goal of preventing leakage on their private dataset. In particular, every example in the training dataset is visible in cleartext to at least one participant. Moreover, their communication complexity is proportional to the dataset size. In contrast, in our setting all data is encrypted and there is no data owner who sees cleartext data examples; furthermore, the communication complexity of our protocol is independent of the dataset size.

The technique of employing *low degree approximation* to speedup computing on encryption data was previously used for other functions, such as $ReLU$, $Sigmoid$ and $Tanh$, see [23,21,25,2,19,8].

2 Preliminaries

In this section we specify standard terminology and notations used throughout this paper, as well as standard definitions for uniform convergence, decision trees, CPA-security, fully homomorphic encryption and privacy-preserving protocols.

2.1 Terminology and notations

We use the following standard notations and terminology. For $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$. A L-dimensional binary vector $y = (y_1, \ldots, y_L)$ is called the *1-hot encoding* of $\ell \in [L]$, if the ℓ'th entry is the only non-zero entry in y.

A function $\mu\colon \mathbb{N} \to \mathbb{R}^+$ is *negligible* in n if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < 1/p(n)$. We use $\mathsf{neg}(\cdot)$ to denote a negligible function if we do not need to specify its name. Unless otherwise indicated, “polynomial” and “negligible” are measured with respect to a system parameter λ called the *security parameter*. We use the shorthand notation PPT for *probabilistic polynomial time* in λ.

A *random variable* A is a function from a finite set S to the non-negative reals with the property that $\sum_{s \in S} A(s) = 1$. A *probability ensemble* $X = \{X(a,n)\}_{a \in \{0,1\}^*, n \in \mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \{0,1\}^*$ and $n \in \mathbb{N}$. Two probability ensembles $X = \{X(a,n)\}_{a \in \{0,1\}^*, n \in \mathbb{N}}$ and $Y = \{Y(a,n)\}_{a \in \{0,1\}^*, n \in \mathbb{N}}$ are said to be *computationally indistinguishable*, denoted by $X \approx_c Y$, if for every non-uniform polynomial-time algorithm $\mathcal{D}$ there exists a negligible function

neg such that for every $a \in \{0,1\}^*$ and every $n \in \mathbb{N}$,

$$|\Pr[\mathcal{D}(X(a,n)) = 1] - \Pr[\mathcal{D}(Y(a,n)) = 1]| \le \mathsf{neg}(n).$$

2.2 Uniform Convergence

We use a standard notion for convergence of functions, as stated next.

Definition 1 (Uniform Convergence). *Let E be a set and let $(f_n)_{n\in\mathbb{N}}$ be a sequence of real-valued functions on E. We say that $(f_n)_{n\in\mathbb{N}}$* uniformly converges *on E to a function f if for every $\epsilon > 0$ there exists a $n_0 \in \mathbb{N}$ such that for all $n \ge n_0$ and $x \in E$ it holds that $|f_n(x) - f(x)| < \epsilon$.*

2.3 Decision Trees

A decision tree T is a binary tree where each internal node corresponds to a partitioning of the input space along one dimension, and each leaf is associated with a label from $\{1, \ldots, L\}$. The decision tree induces a mapping $t\colon \mathbb{R}^k \to \{1, \ldots, L\}$ as follows. A tree T is evaluated on an input sample $x \in \mathbb{R}^k$ by traversing a path in the tree, from root to leaf, using the partitioning rule at each node to decide how to continue; when a leaf is reached, the label associated with it is returned.

The structure of a decision tree is typically learned in order to fit to a given dataset $(\mathcal{X}, \mathcal{Y})$ of n labeled examples for which we ideally want to have: $\forall x \in \mathcal{X} : t(x) = y_x$, i.e., every x is mapped to its corresponding label y_x. The task of finding the optimal tree, that is the tree of a given depth for which the maximal number of the aforementioned equalities hold, is known to be NP-complete [26]. Heuristics used in practice to obtain decision trees given a dataset rely on optimizing the local quality of each partitioning (i.e. each node), by selecting the dimension and threshold value that divide the data into partitions that are each "as pure as possible". The motivation behind this local criterion is that if all data points that arrive to the same leaf have the same label, then by assigning this label to the leaf we are able to categorize this region perfectly. Several measures of purity are commonly used, and we describe the Gini impurity measure in greater detail in Section 3, Figure 2.

2.4 CPA-Secure Public Key Encryption

A public key encryption scheme has the following syntax and correctness requirement.

Definition 2 (Public-Key Encryption (PKE)). *A* public-key encryption (PKE) scheme *with message space $\mathcal{M}$ is a triple* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms satisfying the following conditions:*

- Gen *(key generation) takes as input the security parameter 1^λ, and outputs a pair (pk, sk) consisting of a public key pk and a secret key sk; denoted: $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$.*
- Enc *(encryption) takes as input a public key pk and a message $m \in \mathcal{M}$, and outputs a ciphertext c; denoted: $c \leftarrow \mathsf{Enc}_{pk}(m)$.*
- Dec *(decryption) takes as input a secret key sk and a ciphertext c, and outputs a decrypted message m'; denoted: $m' \leftarrow \mathsf{Dec}_{sk}(c)$.*

Correctness. The scheme is correct *if for every (pk, sk) in the range of* $\mathsf{Gen}(1^\lambda)$ *and every message $m \in \mathcal{M}$,*

$$\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m] = 1.$$

where the probability is taken over the random coins of the encryption algorithm.

A PKE $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is CPA-secure if no PPT adversary $\mathcal{A}$ can distinguish between the encryption of two equal length messages x_0, x_1 of his choice. This is formally stated using the following experiment between a challenger Chal and the adversary $\mathcal{A}$.

The CPA *indistinguishability experiment* $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda)$*:*

1. $\mathsf{Gen}(1^\lambda)$ is run by Chal to obtain keys (pk, sk).
2. Chal provides the adversary $\mathcal{A}$ with pk as well as oracle access to $\mathsf{Enc}_{pk}(\cdot)$, and $\mathcal{A}$ sends to Chal two messages $x_0, x_1 \in \mathcal{M}$ s.t. $|x_0| = |x_1|$.
3. Chal chooses a random bit $b \in \{0, 1\}$, computes a ciphertext $c \leftarrow \mathsf{Enc}_{pk}(x_b)$ and sends c to $\mathcal{A}$. We call c the challenge ciphertext. $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_{pk}(\cdot)$.
4. $\mathcal{A}$ outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$ (0 otherwise).

Definition 3 (CPA-security). *A public key encryption scheme $\mathcal{E}$ =* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has indistinguishable encryptions under chosen-plaintext attacks (or is* CPA-secure*) if for all PPT adversaries $\mathcal{A}$ there exists a negligible function* neg *such that:*

$$\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{neg}(\lambda)$$

where the probability is taken over the random coins of $\mathcal{A}$ and Chal*.*

2.5 Fully Homomorphic Encryption

A fully homomorphic public-key encryption scheme (FHE) is a public-key encryption scheme equipped with an additional PPT algorithm called Eval that supports "homomorphic evaluations" on ciphertexts. For example, given two ciphertexts $c_1 = \mathsf{Enc}(m_1)$ and $c_2 = \mathsf{Enc}(m_2)$ encrypting messages m_1 and m_2 respectively with FHE, it is possible to produce new ciphertexts c_3 and c_4 that decrypt to $\mathsf{Dec}(c_3) = m_1 + m_2$ and $\mathsf{Dec}(c_4) = m_1 \times m_2$ respectively. The correctness requirement is extended to hold with respect to any sequence of homomorphic evaluations performed on ciphertexts encrypted under pk using $\mathsf{Eval}_{pk}(\cdot)$. A fully homomorphic encryption scheme must satisfy an additional property called *compactness* requiring that the size of the ciphertext does not grow with the complexity of the sequence of homomorphic operations. The formal definition follows.

Definition 4 (Public-key FHE scheme [5]). *A homomorphic (public-key) encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *with message space* $\mathcal{M}$ *is a quadruple of PPT algorithms as follows:*

- $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a PKE, as specified in Definition 2.*
- Eval *(homomorphic evaluation) takes as input the public key* pk*, a circuit* $C \colon \mathcal{M}^\ell \to \mathcal{M}$*, and ciphertexts* $c_1, \ldots, c_\ell$*, and outputs a ciphertext* $\widehat{c}$*; denoted:* $\widehat{c} \leftarrow \mathsf{Eval}_{pk}(C, c_1, \ldots, c_\ell)$*.*

Security. A homomorphic encryption scheme is said to be secure if it is a CPA*-secure PKE.*

$\mathcal{C}$*-homomorphism. A homomorphic encryption scheme is* $\mathcal{C}$-homomorphic *for a circuit family* $\mathcal{C}$ *if for all* $C \in \mathcal{C}$ *and for any set of inputs* $x_1, \ldots, x_\ell$ *to* C*, letting* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $c_i \leftarrow \mathsf{Enc}(pk, x_i)$ *it holds that:*

$$\Pr[\mathsf{Dec}_{sk}(\mathsf{Eval}_{pk}(C, c_1, \ldots, c_\ell)) \neq C(x_1, \ldots, x_\ell)] \leq \mathsf{neg}(\lambda)$$

where the probability is taken over all the randomness in the experiment.

Compactness. A homomorphic encryption scheme is compact *if there exists polynomial* $p(\cdot)$ *such that the decryption algorithm can be expressed as a circuit of size* $p(\lambda)$*.*

Fully homomorphic. A homomorphic encryption scheme fully homomorphic*, if it is both compact and* $\mathcal{C}$*-homomorphic for the class* $\mathcal{C}$ *of all efficiently computable circuits.*

2.6 Privacy-Preserving Two-Party Protocols

Next we bring the security definition specifying when a client-server protocol is privacy-preserving against a semi-honest server. A semi-honest server follows the protocol's specifications, albeit it may try to learn additional information.

Security definition. Informally, our protocols are privacy-preserving if the server cannot learn any additional information from participating in the protocol, beyond its output. In our protocols the server has no input or output, so the server should learn nothing from participating in the protocol. The requirement that the server learns nothing can be captured by requiring that the view of the server (i.e., its internal state and received messages during the protocol's execution) in executions on varying (same length) inputs is computationally indistinguishable. This is formalized in [18] Definition 2.6.2 Part 2 for the case of malicious adversaries. We bring their definition, adapted to semi-honest adversaries.

Formally, our protocols involve two-parties, client and server, denoted by Clnt and Srv respectively, where the client has input x and the server has no input (denoted as having input $\perp$), and both have the security parameter λ. The client and server interact in an interactive protocol denoted by $\pi = \langle \mathsf{Clnt}, \mathsf{Srv} \rangle$. An execution of this protocol on client's input x, no server's input, and security paramter λ is denoted by $\langle \mathsf{Clnt}(x), \mathsf{Srv} \rangle$. The server's view during the execution, capturing what the server has learned, is the random variable denoted by $\mathsf{view}^{\pi}_{\mathsf{Srv}}(x, \perp, \lambda)$ and defined by

$$\mathsf{view}^{\pi}_{\mathsf{Srv}}(x, \perp, \lambda) = (r, m_1, \ldots, m_t)$$

where r is the random coins of Srv, and $m_1, \ldots, m_t$ are the messages Srv received during the protocol's execution. The client's output in this execution is denoted by $\mathsf{out}^{\pi}_{\mathsf{Clnt}}(x, \perp, \lambda)$. The protocol is privacy-preserving if the views of the server on (same length) inputs are computationally indistinguishable:

Definition 5 (Privacy-preserving protocol). *An interactive protocol* $\langle \mathsf{Clnt}, \mathsf{Srv} \rangle$ *is* privacy-preserving *for a function* $F : \mathsf{A} \to \mathsf{B}$ *if* Srv *and* Clnt *are PPT machines and there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, the following holds:*

Completeness: *For all* $x \in \mathsf{A}$*,*

$$\Pr[\mathit{out}^{\pi}_{\mathsf{Clnt}}(x, \perp, \lambda) = F(x)] = 1 - \mathsf{neg}(\lambda).$$

Privacy: *For every PPT distinguisher $\mathcal{D}$ that chooses $x_0, x_1 \in \mathsf{A}$ such that $|x_0| = |x_1|$ it holds that:*

$$|\Pr[\mathcal{D}(\mathit{view}^{\pi}_{\mathsf{Srv}}(x_0, \perp, \lambda)) = 1] - \Pr[\mathcal{D}(\mathit{view}^{\pi}_{\mathsf{Srv}}(x_1, \perp, \lambda)) = 1]| \leq \mathsf{neg}(\lambda)$$

where the probability is taken over the random coins of Clnt *and* Srv*.*

3 Decision Trees with Approximated Step Function

In this section we present our algorithms for training and prediction of decision trees. The algorithms are tailored to being evaluated over encrypted data, in the sense of avoiding complexity bottlenecks of homomorphic evaluation.

The key component in our algorithms is a low degree polynomial approximation for the step function "$x < \theta$" (aka, soft-step function); See Section 3.1. The obtained low degree approximation is used to replace the step function at each tree node in our new prediction and training algorithms, presented in Sections 3.2-3.3 respectively.

3.1 Low Degree Approximation of a Step Function

We construct a low-degree polynomial approximating the step function. Specifically, we consider the step function $I_0 \colon \mathbb{R} \to \{0, 1\}$ with threshold zero, defined by: $I_0(x) = 1$ if $x \geq 0$ and $I_0(x) = 0$ otherwise. We aim to replace the step function with a *soft-step function*, *i.e.*, a polynomial approximation.

There are several convenient methods for replacing piece-wise continuous functions with limited-degree polynomial approximation. One approach is to consider the appropriate space of functions as a metric space, and then to find a polynomial of the desired degree that minimizes the deviation from the target function in this metric. Natural choices of metrics are the uniform error, integral square error, and integral absolute error. We opt for the mean square integral solution, that is, the soft-step function would be the solution to the following optimization problem:

$$\phi = \min_{p \in P_n} \int_{-2}^{2} (I_0(x) - p(x))^2 \, dx \qquad (1)$$

where P_n is the set of polynomial functions of degree at most n over the reals. Setting the interval of the approximation to be $[-2, 2]$ is sufficient once we have pre-processed all data to be in the range $[-1, 1]$. A soft-step at $\theta \in [-1, 1]$ is of the form $\phi(x - \theta)$, and thus $x - \theta \in [-2, 2]$.

However, in many cases the sensitivity to error in the approximation is not uniform over the domain. Errors at an interval around the threshold may harm the overall result of the algorithm less, compared to errors away from the threshold value. Adding an importance-weighting of the approximation interval leads to the following optimization problem:

$$\phi = \min_{p \in P_n} \int_{-2}^{2} \left(I_0(x) - p(x)\right)^2 \, w(x) \, dx \tag{2}$$

with a weighting function $w(x) \geq 0 \; \forall x \in [-2, 2]$ and $\int_{-2}^{2} w(x) \, dx = 1$. We note that the unique solution to this problem is obtained by the projection of I_0 onto P_n, in the w-weighted norm (see Chapter II in [34]), or alternatively by applying polynomial regression (minimizing MSE) on a discrete set of points sampled proportionally to w.

Experiments with polynomials which are solutions to Equation 2 with various weight functions and degrees show that by neglecting an interval around the threshold we are able to obtain tighter approximations to the linear phases of the step function. More specifically, by using weighting functions that are zero in a window around the step and constant otherwise, a trade-off is obtained between the slope of the transition and tightness of approximation at the edges of the interval (Figure 1). For larger slopes (smaller neglected windows) the approximation polynomial reaches the $0-1$ plateau faster, but at the price of overshooting and oscillations around the linear parts of the step function. While this can be remedied by choosing very high degree polynomial, computational considerations (especially with FHE arithmetic) lead us to favor smaller degree polynomials.

Approximating in L_∞-norm. The problem of polynomial approximation in L_∞-norm is of great importance in signal processing. The standard way of solving such problems is by applying the Remez Algorithm [32]. In our case, this approach required higher degree polynomials to obtain the same level of accuracy when integrated in our decision tree algorithms, and therefore we chose the L_2 optimization (i.e. the MSE).

3.2 Prediction by Decision Trees with Soft-Step Function

We present our algorithm for decision trees based prediction (see Algorithm 1), which is tailored to achieve efficiency in prediction over encrypted data. Our algorithm is similar to the standard prediction algorithm [31], but where the step function in each node of the decision tree is replaced by its soft-step function counterpart.

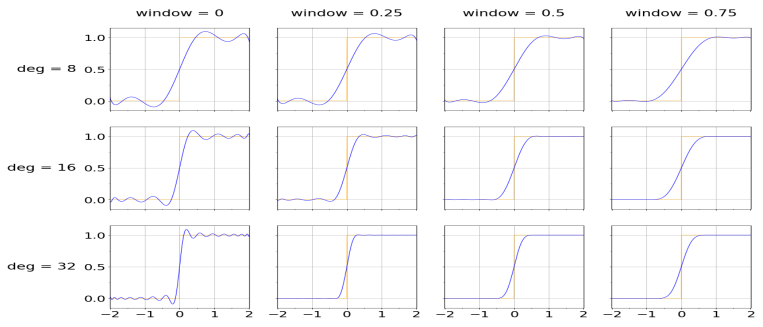


Fig. 1. Polynomial approximations of I_0 in $[-2,2]$, with varying polynomial degrees (rows) and width of neglected window around the transition (columns)

In more detail, both our algorithm and the standard prediction algorithm are similar in the sense that each node v in a decision tree T is associated with a threshold $v.\theta \in \mathbb{R}$ and a feature $v.feature \in [k]$, and the prediction is a label $\ell \in \{1,\ldots,L\}$. The difference between the algorithms is specified next.

In the standard prediction algorithm, each new sample $x \in \mathbb{R}^k$ is associated with a single path from root to leaf, where the path continues to the right-child if the step function $I_{v.\theta}(x[v.feature])$ evaluates to 1 and to the left-child otherwise. We denote by T_x the path in T associated with x. The prediction for x is the label associated with the leaf in T_x.

In our prediction algorithm (see Algorithm 1), we replace the step function $I_{v.\theta}(x[v.feature])$ by the soft-step function $\phi\big(x[v.feature]-v.\theta\big)$, where ϕ is obtained via Equation 2. Note that by using a soft-step function we no longer traverse a single path in the tree. Instead, our algorithm traverses all paths in the tree and computes a weighted combination of all the leaves values, where each leaf value is the 1-hot encoding of the label associated with the leaf. The output is a length L vector assigning a likelihood score to each label, which is in turn interpreted as outputting the label with the highest score.

We make a few remarks regarding extensions of Algorithm 1. a) The algorithm straightforwardly extends to the case where each leaf is associated with a vector of labels' likelihoods $leaf\_value \in \mathbb{R}^L$ (rather than a 1-hot encoding of a label). b) The algorithm naturally extends to evaluation of base-classifiers for random forests and tree boosting, hence our method supports these common tree-based machine learning methods as well. c) Finally we note that the algorithm can also be viewed as a "recipe" for converting an existing tree model to a polynomial form of low degree, which may be of independent interest.

Algorithm 1 Tree Prediction

Let T be a decision tree, where each node v is a data structure containing the following fields: $v.feature$ and $v.\theta$ holding the feature and threshold associated with v; $v.leaf\_value$ holding the 1-hot encoding of the label associated with v in case v is a leaf; and $v.right$ and $v.left$ for the right and left children of v, respectively. The variable v is initialized to be the root of T.
We denote by ϕ the soft-step function from Equation 2 in Section 3.1.
Input: $x \in [-1,1]^k$, where k is the number of features.

```
function Tree_Predict($v$, $x$)
    if $v$ is a leaf then
        return $v.leaf\_value$
    else
        return $\phi\big(x[v.feature] - v.\theta\big) \cdot$ Tree_Predict($v.right, x$) +
                $\phi\big(v.\theta - x[v.feature]\big) \cdot$ Tree_Predict($v.left, x$)
    end if
end function
```

We next show that Algorithm 1 outputs the same prediction as the standard algorithm (aka, it is *correct*), whenever ϕ is sufficiently close to I_0. Elaborating on the above, essentially we show that if $(\phi_n)_{n\in\mathbb{N}}$ uniformly converges to the step function I_0, then for all sufficiently large n, Algorithm 1, when executed with ϕ_n as the soft-step function, assigns more than half the weight to the leaf associated with path that is chosen by the standard algorithm, and so the outputted label is identical to the label in the standard algorithm. This is a bit over simplified though: I_0 has a discontinuity point at 0 implying that no family of polynomials can uniformly converge to I_0 on an interval containing 0, say, $[-2,2]$. Nonetheless, for every $\delta > 0$, there exists a family of polynomials $(\phi_n)_{n\in\mathbb{N}}$ that uniformly converges to I_0 on the punctured interval $[-2,2] \setminus (-\delta,\delta)$; in particular this holds for the polynomials computed by Remez Algorithm.[5] Employing such polynomials we guarantee correctness (i.e., the output of Algorithm 1 and the standard algorithm are the same) for all samples that are δ-far from the discontinuity point.

[5] In our implementation and evaluation of Algorithm 1, rather than using the polynomials computed by Remez Algorithm, we use the solution to the weighted mean square error (weighted MSE) optimization problem specified in Eq 2. We use the latter because our experiments indicate that it achieves a better accuracy vs. degree tradeoff; that is, for comparable accuracy the weighted MSE solution required in our experiments a lower degree than the polynomial produced by Remez Algorithm.

Theorem 1 (Correctness). *Let $\delta > 0$ and let $(\phi_n)_{n\in\mathbb{N}}$ be a sequence of functions that uniformly converges to I_0 on $[-2,2]\setminus(-\delta,\delta)$. For every tree T of depth d, there exists $n_0 = n_0(d)$ such that for all $n \geq n_0$, the following holds. Algorithm 1, instantiated with ϕ_n and T, is correct on all samples x that satisfy $\min_{v\in\mathsf{T}} |x[v.feature] - v.\theta| > \delta$.*

Proof. Let $\delta > 0$, and let $(\phi_n)_{n\in\mathbb{N}}$ be a sequence of functions that uniformly converges to I_0 on $[-2,2]\setminus(-\delta,\delta)$. Fix a tree T, and let d denote its depth and L denote the labels associated with T. For a sample x we denote by $\mathsf{T}_x = (v_0^*,\ldots,v_d^*)$ the path in T traversed by the standard algorithm on input x, and denote by ℓ^* the label outputted by it (i.e., the label associated with the leaf of T_x). We show that for all sufficiently large n (that depend only on d), Algorithm 1 instantiated with ϕ_n and T, outputs ℓ^* on all samples x with $\min_{v\in\mathsf{T}} |x[v.feature] - v.\theta| > \delta$.

For this purpose first observe that for every sample x, the weight assigned by Algorithm 1 to each path $\mathsf{P} = (v_0,\ldots,v_d)$ from root to leaf in T, denoted by $w(\mathsf{P})$, is the product of the weights assigned by the algorithm to each internal node v_i on the path, where the weight of each node v_i is equal to $\phi_n(x[v_i.feature] - v_i.\theta)$ if the path continues from v_i to its right child, and it is $\phi_n(v_i.\theta - x[v_i.feature])$ otherwise. That is,

$$w(\mathsf{P}) = \prod_{i=0}^{d-1} \phi_n\left((x[v_i.feature] - v_i.\theta)\cdot \mathsf{isRC}(v_i,v_{i+1})\right). \tag{3}$$

where $\mathsf{isRC}(v_i,v_{i+1}) = 1$ if v_{i+1} is a right-child of v_i, and -1 otherwise.

Next we analyze the weight $w(\mathsf{P})$ assigned by Algorithm 1 to each path P from root to leaf. In Lemma 1 we show that for all sufficiently large $n = n(d)$, and all $x \in \mathbb{R}^k$ s.t. $\min_{v\in\mathsf{T}} |x[v.feature] - v.\theta| > \delta$, it holds that most of the weight is assigned to the path T_x, that is,

$$w(\mathsf{T}_x) > \sum_{\mathsf{P}=(v_0,\ldots,v_d)\neq\mathsf{T}_x} w(\mathsf{P}). \tag{4}$$

Finally we derive from Eq 4 the desired conclusion that Algorithm 1 is correct by showing that it assigns the highest score to the label ℓ^* outputted by the standard algorithm. For this purpose first recall that leaf values are the 1-hot encoding of the label associated with the leaf. In particular, for the leaf v_d^* in the path T_x, the associated label is ℓ^* and therefore:

$$v_d^*.leaf\_value[\ell^*] = 1.$$

Next observe that the score assigned to the label ℓ^* by Algorithm 1 is the sum of weights over all paths terminating in leaves associated with this label. In particular, this score is at least as large as the weight $w(\mathsf{T}_x)$. That is,

$$\begin{aligned}\sum_{\mathsf{P}=(v_0,\ldots,v_d)\in\mathsf{T}} & w(\mathsf{P})\cdot v_d.leaf\_value[\ell^*] \\ &\geq w(\mathsf{T}_x)\cdot v_d^*.leaf\_value[\ell^*] \\ &= w(\mathsf{T}_x)\end{aligned} \tag{5}$$

Moreover, for every label $\ell \in [L]$ such that $\ell \neq \ell^*$ it holds that

$$w(\mathsf{T}_x)\cdot v_d^*.leaf\_value[\ell] = 0.$$

Therefore, the score assigned to ℓ by Algorithm 1 is upper bounded by the sum of weights over all paths other than T_x. That is, the score of ℓ is upper bounded by:

$$\begin{aligned}&\sum_{\mathsf{P}=(v_0,\ldots,v_d)\in\mathsf{T}} w(\mathsf{P})\cdot v_d.leaf\_value[\ell] \\ &= \sum_{\mathsf{P}=(v_0,\ldots,v_d)\in\mathsf{T}\ s.t.\ P\neq\mathsf{T}_x} w(\mathsf{P})\cdot v_d.leaf\_value[\ell] \\ &\leq \sum_{\mathsf{P}=(v_0,\ldots,v_d)\in\mathsf{T}\ s.t.\ P\neq\mathsf{T}_x} w(\mathsf{P}) \\ &< w(\mathsf{T}_x)\end{aligned} \tag{6}$$

(where the inequality before last follows from $v_d.leaf\_value[\ell]$ being in $\{0,1\}$, and the last inequality follows from Equation 4). We conclude that the score assigned by Algorithm 1 to ℓ^* is strictly larger than the score Algorithm 1 assigns to any other label ℓ, which implies that

$$\ell^* = \arg\max_{\ell\in[L]} \sum_{\mathsf{P}=(v_0,\ldots,v_d)\in\mathsf{T}} w(\mathsf{P})\cdot v_d.leaf\_value[\ell] \tag{7}$$

as desired. □

Lemma 1. *Let $\delta > 0$ and let $(\phi_n)_{n\in\mathbb{N}}$ be a sequence of functions that uniformly converges to I_0 on $[-2,2]\setminus(-\delta,\delta)$. For every tree T of depth d, there exists $n_0 = n_0(d)$ such that for all $n \geq n_0$, the following holds. Algorithm 1, when instantiated with ϕ_n and T, satisfies that*

$$w(\mathsf{T}_x) > \sum_{\mathsf{P}=(v_0,\ldots,v_d)\neq\mathsf{T}_x} w(\mathsf{P})$$

for all samples x s.t. $\min_{v\in\mathsf{T}} |x[v.feature] - v.\theta| > \delta$.

Proof. The proof idea is as follows. If we were using in Algorithm 1 the step-function I_0, rather than ϕ_n, then we would get that the weight $w(\mathsf{T}_x)$ is one whereas the weight $w(\mathsf{P})$ is zero for any path $\mathsf{P} \neq \mathsf{T}_x$ from root to leaf. Now, by taking sufficiently large n, we can ensure that $\phi_n(z)$ is as close as we'd like to $I_0(z)$ on all $z \notin (-\delta, \delta)$. In particular, we can enforce $w(\mathsf{T}_x) > 1/2$ and $\sum_{\mathsf{P}\neq\mathsf{T}_x} w(\mathsf{P}) < 1/2$ as long as $|x[v.feature] - v.\theta| > \delta$ for all $v \in \mathsf{T}$. In this case indeed $w(\mathsf{T}_x) > \sum_{\mathsf{P}\neq\mathsf{T}_x} w(\mathsf{P})$. We proceed with the formal details.

Fix $\delta > 0$ and let T be a tree of depth d. Set $\epsilon = \epsilon(d) = \min\{1 - 2^{-1/d}, 2^{-2d}\}$. The premise of uniform convergence means that there exists $n_0 = n_0(\epsilon) = n_0(d)$ such that for all $n > n_0$, the following holds:

$$|\phi_n(z) - I_0(z)| < \epsilon \text{ for all } z \in [-2, 2] \setminus (-\delta, \delta).$$

Since $\min_{v\in\mathsf{T}} |x[v.feature] - v.\theta| > \delta$, the above implies that for all $v \in \mathsf{T}$,

$$|\phi_n\left(x[v.feature] - v.\theta\right) - I_0\left(x[v.feature] - v.\theta\right)| < \epsilon \tag{8}$$

and likewise $|\phi_n\left(v.\theta - x[v.feature]\right) - I_0\left(v.\theta - x[v.feature]\right)| < \epsilon$.

We first lower bound the weight of the path $\mathsf{T}_x = (v_0^*, \ldots, v_d^*)$ traversed by the standard algorithm. For all nodes $v_i^* \in \mathsf{T}_x$, it holds by the definition of T_x that $I_0\left((x[v_i^*.feature] - v_i^*.\theta) \cdot \mathsf{isRC}(v_i^*, v_{i+1}^*)\right) = 1$, implying by Eq 8 that the weight assigned to T_x by Algorithm 1 is at least:

$$w(\mathsf{T}_x) = \prod_{i=0}^{d-1} \phi_n\left((x[v_i^*.feature] - v_i^*.\theta) \cdot \mathsf{isRC}(v_i^*, v_{i+1}^*)\right) > (1-\epsilon)^d.$$

Assigning $\epsilon \leq 1 - 2^{-1/d}$ we get that:

$$w(\mathsf{T}_x) > 1/2. \tag{9}$$

We next upper bound the sum of weights over all paths other than T_x. Let $\mathsf{P} = (v_0, \ldots, v_d)$ be a path from root to leaf in T such that $\mathsf{P} \neq \mathsf{T}_x$. Let $i \in \{1, \ldots, d\}$ be the first index where $v_i \neq v_i^*$. Namely, in v_{i-1} P proceeds to a different child than T_x implying that:

$$I_0\left((x[v_{i-1}.feature] - v_{i-1}.\theta) \cdot \mathsf{isRC}(v_{i-1}, v_i)\right) = 0.$$

By Equation 8, the above implies that:

$$\phi_n\left((x[v_{i-1}.feature] - v_{i-1}.\theta) \cdot \mathsf{isRC}(v_{i-1}, v_i)\right) < \epsilon.$$

Furthermore, for any other internal node $v_j \in \mathsf{P}$ $(j \neq i-1)$, since I_0 is upper bounded by 1, then by Equation 8 it holds that:

$$\phi_n\left((x[v_j.feature] - v_j.\theta) \cdot \mathsf{isRC}(v_j, v_{j+1})\right) < 1 + \epsilon.$$

Put together, the product of all these values is upper bounded by:

$$w(\mathsf{P}) < \epsilon \cdot (1+\epsilon)^{d-1}.$$

The total weight of all paths in T besides T_x is therefore upper bounded by:

$$\sum_{\mathsf{P} \neq \mathsf{T}_x} w(\mathsf{P}) < (2^d - 1) \cdot \epsilon \cdot (1+\epsilon)^{d-1} < 2^{2d-1} \cdot \epsilon \leq 1/2 \tag{10}$$

where the last two inequalities hold since $\epsilon < 1$ and $\epsilon \leq 2^{-2d}$ respectively. This concludes the proof, as desired. □

3.3 Training Decision Trees with Soft-Step Function

The standard training procedure considers splits that partition the training dataset $\mathcal{X}$ and builds a tree according to local objectives based on the number of examples of each label that flow to each side at a chosen, per node, split. In the training procedure, at each node, impurity scores are calculated for each potential split, then the feature and threshold are chosen in order to minimize some impurity measure, *e.g.*, the weighted Gini impurity (see Figure 2).

Traditionally, the training procedure associates with each node a set of indicators $W = \{w_x\}_{x \in \mathcal{X}}$, so that w_x is a bit indicating if example x is participating in the training of a sub-tree rooted at this node, and W is updated for the children nodes as follows: for the chosen feature i^*, and threshold θ^*, the right sub-tree (respectively, left sub-tree)

$$\forall x \in \mathcal{X} : w_x^{\mathsf{right}} = w_x \cdot I_{\theta^*}(x[i^*]) \quad \left(\text{resp. } w_x^{\mathsf{left}} = w_x \cdot (1 - I_{\theta^*}(x[i^*]))\right) \tag{11}$$

In our approach, to avoid the comparison operation that is expensive over encrypted data, we replace the step function I_0 by the low-degree polynomial approximation ϕ obtained via Equation 2.

Notice that our approximated version of Equation 11 has real valued weights instead of Boolean indicators. This means that *every example* reaches every node, and is evaluated at all nodes. This results in a soft partition of the data, where the two children nodes get each data point,

Algorithm 2 Tree Training

We denote by $(\mathcal{X}, \mathcal{Y})$ the input dataset, and by $W = \{w_x\}_{x \in \mathcal{X}}$ a set of real valued weights initialized to 1, associating a weight w_x with each example $x \in [-1,1]^k$ in $\mathcal{X}$ at each node of the tree. We denote by k and L the number of features and labels in the associated problem, respectively. We denote by S the set of considered thresholds. The parameter $\mathsf{maximal\_depth}$ is the depth of the trained tree, the variable *depth* is initialized to 0. We denote by ϕ the soft-step function from Equation 2. The function $\mathsf{Gini}(\cdot)$ in Figure 2 computes the weighted Gini impurity and returns the best threshold and feature.

Input: A set $\mathcal{X}$ of n examples and the set $\mathcal{Y}$ of corresponding labels, where each example $x \in \mathcal{X}$ is in $[-1,1]^k$ and the corresponding $y_x \in \mathcal{Y}$ is a 1-hot encoding of the label of x.

Output: A full binary tree $\mathsf{T} = (V, E)$ of depth $\mathsf{maximal\_depth}$, where each internal node $v \in V$ is a data structure containing the following fields: $v.feature$ and $v.\theta$ holding the feature and threshold associated with v; $v.leaf\_value$ holding the label associated with v in 1-hot encoding in case v is a leaf; and $v.right$ and $v.left$ for the right and left children of v, respectively. The variable v is initialized to be the root of T.

```
1: function Tree_Train($(\mathcal{X}, \mathcal{Y}), W, depth, v$)
2:     if reached maximal_depth then                                    ▷ leaf node
3:         $v.leaf\_value \leftarrow \arg\max_{\ell \in [L]} \sum_{x \in \mathcal{X}} w_x \cdot y_x$ (in 1-hot encoding)
4:     else                                              ▷ search best split for this node
5:         for each feature $i$ and each threshold $\theta$ do
6:             $\mathsf{right}[i,\theta] \leftarrow \sum_{x \in \mathcal{X}} w_x \cdot \phi(x[i] - \theta) \cdot y_x$
7:             $\mathsf{left}[i,\theta] \leftarrow \sum_{x \in \mathcal{X}} w_x \cdot \phi(\theta - x[i]) \cdot y_x$
8:         end for
9:         $(v.feature, v.\theta) \leftarrow \mathsf{Gini}(\{\mathsf{right}[i,\theta], \mathsf{left}[i,\theta]\}_{i \in [k], \theta \in S})$
10:                                                                  ▷ See Figure 2
11:        $\forall x \in \mathcal{X} : w_x^{\mathsf{right}} \leftarrow w_x \cdot \phi(x[i^*] - \theta^*)$
12:        Tree_Train($(\mathcal{X}, \mathcal{Y}), \{w_x^{\mathsf{right}}\}_{x \in \mathcal{X}}, depth + 1, v.right$)
13:                                                      ▷ build right-side sub-tree
14:        $\forall x \in \mathcal{X} : w_x^{\mathsf{left}} \leftarrow w_x \cdot \phi(\theta^* - x[i^*])$
15:        Tree_Train($(\mathcal{X}, \mathcal{Y}), \{w_x^{\mathsf{left}}\}_{x \in \mathcal{X}}, depth + 1, v.left$)
16:                                                       ▷ build left-side sub-tree
17:    end if
18: end function
```

weighted differently, rather than hard split of the data. In order to efficiently keep track of the weight of each data example at each node, we keep a weights set W while constructing the tree during training. All weights are initialized to 1, and recursively multiplied by the polynomial approximation at the current node before passing on to the children nodes. The details of the training algorithm are presented in Algorithm 2.

Looking ahead, we carefully divide the operations in Algorithm 2 to those where homomorphic evaluation on encrypted data is "efficient" vs. "costly". Concretely, addition, multiplication and evaluating the low de-

Procedure: Gini impurity computation.
We denote by k and L the number of features and labels in the associated problem, respectively. We denote by S the set of considered thresholds. The function computes the weighted Gini impurity and returns the best threshold and feature.
Given a set of L-dimensional vectors $\{\mathsf{right}[i,\theta], \mathsf{left}[i,\theta]\}_{i\in[k],\theta\in S}$ proceed as follows:

1. For each threshold $\theta \in S$, each feature $i \in [k]$, and $side \in \{\mathsf{right}, \mathsf{left}\}$ compute

$$\mathsf{total\_side}[i,\theta] \leftarrow \sum_{\ell\in L} \mathsf{side}[i,\theta][\ell]$$

$$\tilde{I}_G[i,\theta] = \sum_{side\in\{\mathsf{right},\mathsf{left}\}} \left(1 - \sum_{\ell\in L} \left[\frac{\mathsf{side}[i,\theta][\ell]}{\mathsf{total\_side}[i,\theta]}\right]^2\right) \cdot \mathsf{total\_side}[i,\theta]$$

2. Return the selected feature and threshold $(i^*, \theta^*) \leftarrow \arg\min_{i,\theta} \tilde{I}_G[i,\theta]$

Fig. 2. The weighted Gini impurity computation

gree polynomial ϕ are efficient, whereas computing the Gini Impurity (Figure 2) involves the costly division and argmin operations.

4 Prediction and Training on Encrypted Data

In this section we present our secure protocols for prediction and training of tree based models where both the training dataset and the data for prediction are encrypted with FHE. The protocols are an adaptation of Algorithms 1–2 in Section 3 to interactive settings and where data is encrypted throughout the computation. See our protocol for prediction over encrypted data in Figures 3–4 in Section 4.1, and our protocol for training over an encrypted dataset in Figures 5–6 in Section 4.2.

4.1 Decision-Tree based Prediction on Encrypted Data

We present our privacy preserving protocol for decision tree based prediction. The protocol is between a client holding a data sample and a server holding a tree, and consists of the following steps: the client encrypts the data sample and sends the encrypted sample to the server; the server homomorphically evaluates Algorithm 1 on the encrypted sample and sends the encrypted outcome to the client; the client decrypts to obtain the prediction. The protocol is privacy preserving for Algorithm 1; see Definition 5 and Theorem 2. The client's and communication complexity are proportional to the size of the encrypted input and output, while being independent of the tree size and depth; the server's complexity is linear in the tree size and the number of labels and polynomial in the security parameter.

Our protocol can be executed on any tree based model such as Random Forest or Boosted Tree algorithms. Our protocol extends to the case where both the sample and the decision tree are encrypted, providing secrecy for both the data sample and the tree.

See the detail of our protocol on encrypted sample and cleartext trees in Figures 3–4; the extension to encrypted trees below; and the privacy and complexity analysis in Theorem 2.

Shared parameters: An FHE encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$; a security parameter λ; a soft-step function ϕ obtained via Equation 2; the numbers of features k and labels L, and a decision tree T.
In the tree T, each internal node v is a data structure containing four fields: $v.feature$ and $v.\theta$ denoting the feature and threshold associated with v, and $v.right$ and $v.left$ denoting its right and left children; each leaf v consists of the field $v.leaf\_value \in \mathbb{R}^L$ holding the 1-hot encoding of the label associated with v.

Client's input: A normalized data sample $x \in [-1, 1]^k$.

Client's output: A label $\ell \in L$.

Server's input and output: The server has no input and receives no output.

The protocol $\mathsf{PP} = \langle \mathsf{Clnt}_{\mathsf{PP}}, \mathsf{Srv}_{\mathsf{PP}} \rangle$ proceeds as follows:

1. **Input outsourcing phase:**
   (a) $\mathsf{Clnt}_{\mathsf{PP}}$ runs $\mathsf{Gen}(1^\lambda)$ to obtain a public and secret key pair (pk, sk).
   (b) $\mathsf{Clnt}_{\mathsf{PP}}$ encrypts each entry in x to obtain a vector of ciphertexts $\mathbf{c_x}$, and sends $\mathbf{c_x}$ together with pk to $\mathsf{Srv}_{\mathsf{PP}}$.
2. **Computation phase:** $\mathsf{Srv}_{\mathsf{PP}}$ recursively computes the subroutine $\mathsf{Enc\_Predict}(v, \mathbf{c_x})$ in Figure 4 starting from the root node. Let **res** be the recursion result, *i.e.*, a L-dimensional vector of ciphertexts returned at the root.
3. **Output phase:** $\mathsf{Srv}_{\mathsf{PP}}$ sends **res** to $\mathsf{Clnt}_{\mathsf{PP}}$, who decrypts it and outputs $label \leftarrow \arg\max_{\ell \in L} \mathsf{p\_res}_\ell$, where $\mathsf{p\_res}$ is the decryption of **res**.

Fig. 3. The prediction protocol $\mathsf{PP} = \langle \mathsf{Clnt}_{\mathsf{PP}}, \mathsf{Srv}_{\mathsf{PP}} \rangle$ for decision trees. The sever homomorphically evaluates a weighted sum of the leaf values to obtain an encrypted vector of labels' scores. The client decrypts and outputs the label with highest score.

Extension to prediction on encrypted data and encrypted tree. For certain applications it is required that the tree used for prediction remains hidden from the server. Minor modification to the protocol in Figure 3 transform it to a protocol that keeps both the data sample and the tree private. Essentially, the tree will be transmitted encrypted to

Subroutine $\mathsf{Enc\_Predict}(v, \mathbf{c_x})$ where v is a node in T, and $\mathbf{c_x}$ is a vector of L ciphertexts.

1. If v is not a leaf, homomorphically evaluate the following formula (using Eval_{pk}) and return the resulting ciphertext:

$$\phi\big(\mathbf{c_x}[v.feature] - v.\theta\big) \cdot \mathsf{Enc\_Predict}(v.right, \mathbf{c_x}) +$$
$$\phi\big(v.\theta - \mathbf{c_x}[v.feature]\big) \cdot \mathsf{Enc\_Predict}(v.left, \mathbf{c_x})$$

2. Otherwise return $v.leaf\_value$.

Fig. 4. The subroutine $\mathsf{Enc\_Predict}(\cdot,\cdot)$ operates recursively on a node and ciphertext pair. The subroutine is an adjustment of Algorithm 1 to operate over encrypted data.

the server (not necessarily by the client), and the protocol in Figure 3 is modified to be executed on encrypted sample and encrypted tree.

In detail, the tree T is encrypted as follows. For each node v in T, the field $v.feature$ is first transformed into a 1-hot encoding (i.e., a binary vector of dimension k with a single non-zero entry at the index specified by the feature). Then the fields $v.feature$ (in the 1-hot encoding), $v.\theta$ and $v.leaf\_value$ are encrypted with the FHE using the public key pk (generated in Figure 3, Step 1a); denote the resulting ciphertexts by $\tilde{\mathsf{v}}.\mathsf{feature}$, $\tilde{\mathsf{v}}.\theta$ and $\tilde{\mathsf{v}}.\mathsf{leaf\_value}$ respectively. The fields $v.right$ and $v.left$ are not encrypted as they are not secret. The protocol is modified as follows. In Figure 4, Step 1, instead of using $v.feature$ to directly access the desired entry in $\mathbf{c_x}$, we use the encrypted 1-hot encoding $\tilde{\mathsf{v}}.\mathsf{feature}$; specifically, we replace in the homomorphically evaluated formula each $\mathbf{c_x}[v.feature]$ by $\sum_{i\in[k]} \mathbf{c_x}[i] \cdot \tilde{\mathsf{v}}.\mathsf{feature}[i]$. Finally, in Figure 4, Step 2, instead of returning the cleartext value $v.leaf\_value$ we return the encrypted value $\tilde{\mathsf{v}}.\mathsf{leaf\_value}$.

This extension preserves privacy of both the sample x and the tree T. Moreover, the complexity is similar to the original protocol: the server only computes k additional multiplications and additions per node, for k the number of features, and the overall multiplicative depth of the homomorphic evaluation grows only by 1.

Theorem 2 (Privacy-preserving prediction). $\mathsf{PP} = \langle \mathsf{Clnt_{PP}}, \mathsf{Srv_{PP}} \rangle$ *(Figure 3) is a privacy-preserving protocol for Algorithm 1 provided that the underlying encryption scheme $\mathcal{E}$ is CPA-secure. Moreover, the computation phase (Figure 3 Step 2) is non-interactive; the complexity of* $\mathsf{Srv_{PP}}$ *is $O(m \cdot L) \cdot poly(\lambda)$, where m denote the number of decision tree nodes, L the number of labels, and λ is the security parameter; the complexity of* $\mathsf{Clnt_{PP}}$ *is proportional to encrypting the input and decrypting the output.*

Proof (of Theorem 2). Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ denote the FHE encryption scheme used in the protocol $\mathsf{PP} = \langle \mathsf{Clnt}_{\mathsf{PP}}, \mathsf{Srv}_{\mathsf{PP}} \rangle$, and suppose $\mathcal{E}$ is CPA-secure. To prove that the protocol is privacy-preserving we prove it is PPT, complete and private. Furthermore, we analyze its complexity.

We first analyze the complexity of PP and prove it is PPT. $\mathsf{Clnt}_{\mathsf{PP}}$ given input $x \in [-1,1]^k$ performs the following operations: a single execution of Gen, k executions of Enc (one for each features in x), L executions of Dec (one for each label weight in the output result $\mathsf{p\_res}$), and computing the maximum of the resulting L values. Since $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ all have complexity $poly(\lambda)$, we conclude that $\mathsf{Clnt}_{\mathsf{PP}}$ is PPT and its complexity is proportional to encrypting the input and decrypting the output. Our $\mathsf{Srv}_{\mathsf{PP}}$ performs a constant number of basic homomorphic operations (i.e. additions and multiplications) for each internal node, and $O(L)$ for each leaf, where each basic homomorphic operation has complexity $poly(\lambda)$. We conclude that $\mathsf{Srv}_{\mathsf{PP}}$ is PPT with complexity $O(m \cdot L) \cdot poly(\lambda)$.

We next prove that $\mathsf{PP} = \langle \mathsf{Clnt}_{\mathsf{PP}}, \mathsf{Srv}_{\mathsf{PP}} \rangle$ is complete. Observe that PP homomorphically evaluates the same function as computed in Algorithm 1, and hence completeness follows immediately from the correctness of $\mathcal{E}$.

Finally, we prove that PP satisfies the privacy condition of Definition 5. Assume by contradiction that privacy does not hold for PP. That is, there exists a PPT distinguisher $\mathcal{D}$ that chooses $x_0, x_1 \in \mathsf{A}$ with $|x_0| = |x_1|$, and a polynomial $p(\cdot)$ such that for infinitely many $\lambda \in \mathbb{N}$:

$$\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{PP}}_{\mathsf{Srv}_{\mathsf{PP}}}(x_1, \perp, \lambda)) = 1] - \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{PP}}_{\mathsf{Srv}_{\mathsf{PP}}}(x_0, \perp, \lambda)) = 1] \geq p(\lambda) \tag{12}$$

We show that given $\mathcal{D}$ we can construct an adversary $\mathcal{A}$ that violate the CPA security of $\mathcal{E}$. The adversary $\mathcal{A}$ participates in $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}$ as follows:

1. Upon receiving pk output x_0, x_1.
2. Upon receiving $\mathbf{c}_x \leftarrow \mathsf{Enc}_{pk}(x_b)$ behave exactly as $\mathsf{Srv}_{\mathsf{PP}}$ behaves while executing PP upon receiving $(\mathbf{c}_x, pk)$ from $\mathsf{Clnt}_{\mathsf{PP}}$.
3. Run the distinguisher $\mathcal{D}$ on $\mathsf{view}^{\mathsf{PP}}_{\mathsf{Srv}_{\mathsf{PP}}}$ ($\mathsf{Srv}_{\mathsf{PP}}$'s view in $\mathcal{A}$ during step 2) and output whatever $\mathcal{D}$ outputs.

The adversary $\mathcal{A}$ is PPT due to x_0, x_1 being efficiently samplable and $\mathsf{Srv}_{\mathsf{PP}}$ and $\mathcal{D}$ being PPT. We denote by $\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv}_{\mathsf{PP}}}(x_{b^*}, \perp, \lambda)$ the view of $\mathsf{Srv}_{\mathsf{PP}}$, simulated by $\mathcal{A}$, in the execution of $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}$ with bit b^* being selected by the challenger. Since $\mathcal{A}$ behaves exactly as $\mathsf{Srv}_{\mathsf{PP}}$ in PP, it holds that for every $b^* \in \{0,1\}$,

$$\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{PP}}_{\mathsf{Srv}_{\mathsf{PP}}}(x_{b^*}, \perp, \lambda)) = 1] = \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv}_{\mathsf{PP}}}(x_{b^*}, \perp, \lambda)) = 1] \tag{13}$$

From Equations 12 and 13 it follows that:

$$\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{PP}}}(x_1, \perp, \lambda)) = 1] - \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{PP}}}(x_0, \perp, \lambda)) = 1] \geq p(\lambda) \tag{14}$$

Therefore, we obtain that:

$$\begin{aligned}
&\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1] \\
&\quad = \frac{1}{2} \cdot \left(\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1 | b = 1] + \Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1 | b = 0] \right) \\
&\quad = \frac{1}{2} \cdot \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{PP}}}(x_1, \perp, \lambda)) = 1] + \frac{1}{2} \cdot \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{PP}}}(x_0, \perp, \lambda)) = 0] \\
&\quad = \frac{1}{2} + \frac{1}{2} \cdot \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{PP}}}(x_1, \perp, \lambda)) = 1] - \frac{1}{2} \cdot \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{PP}}}(x_0, \perp, \lambda)) = 1] \\
&\quad \geq \frac{1}{2} + \frac{1}{2} \cdot p(\lambda)
\end{aligned}$$

where the last inequality follows from Equation 14. Combining this with $\mathcal{A}$ being PPT we derive a contradiction to $\mathcal{E}$ being CPA secure. This concludes the proof. $\square$

4.2 Decision-Tree Training on Encrypted Data

In this section we present our privacy-preserving protocol for training decision trees and Random Forests. Our protocol is a careful adaptation of Algorithm 2 to the client-server setting in a way that the computational burden is almost fully on the server: the computation phase (Figure 5 Step 2) attains client and communication complexity that are independent of the training dataset size, whereas only the server's complexity grows with the training dataset size. The only dependence of the client in the dataset size is when the data is encrypted and uploaded to the sever (Figure 5 Step 1). See Figures 5-6 and Theorem 3.

The computation phase of our protocol (Figure 5 Step 2) is an homomorphic evaluation of Algorithm 2 but with client's aid for few and lightweight computations. Concretely, we replace all operations on cleartext data with their corresponding homomorphic operations on encrypted data, except for computing the impurity score (Line 9 in Algorithm 2) that is done with the aid of the client. To compute the impurity score, the server first homomorphically aggregates encrypted data samples from the training set into $|S| \cdot k \cdot L$ ciphertexts and sends these encrypted aggregates to the client (where $|S|$ is the number of considered thresholds, k the number of features, and L the number of labels). Next, the client decrypts these aggregates and computes the impurity score on the resulting cleartext values (Figure 2) to obtain the chosen threshold and feature. The client then encrypts the chosen threshold and feature and sends the ciphertexts to the server. This procedure attains the desired complexity and privacy: the client's complexity is independent of the dataset size, and the server is exposed only to encrypted values; See Theorem 3.

We remark that although the protocol in Figure 5 is presented with respect to the Gini impurity measure, it can be instantiated with other standard impurity measures, such as *entropy*. Furthermore, the leaves of T can be post-processed by the client to associate a single label with each leaf instead of a likelihood score vector.

Theorem 3 (Privacy-preserving training). $\mathsf{TP} = \langle \mathsf{Clnt}_{\mathsf{TP}}, \mathsf{Srv}_{\mathsf{TP}} \rangle$ *(Figure 5) is a privacy-preserving protocol for Algorithm 2 provided that the underlying encryption scheme $\mathcal{E}$ is CPA-secure. Moreover, the computation phase (Figure 5 Step 2) is a d-round protocol for depth d trees, with* $\mathsf{Clnt}_{\mathsf{TP}}$ *and communications complexity $O(m \cdot k \cdot |S| \cdot L) \cdot poly(\lambda)$, and* $\mathsf{Srv}_{\mathsf{TP}}$ *complexity $O(n \cdot m \cdot |S| \cdot k \cdot L) \cdot poly(\lambda)$, where n is the training set size, $k, L, m, |S|$ denote the number of features, labels, decision tree nodes and considered thresholds, respectively, and λ is the security parameter.*

Proof (of Theorem 3). Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ denote the FHE encryption scheme used in the protocol $\mathsf{TP} = \langle \mathsf{Clnt}_{\mathsf{TP}}, \mathsf{Srv}_{\mathsf{TP}} \rangle$, and suppose $\mathcal{E}$ is CPA-secure. To prove that the protocol is privacy-preserving we prove it is PPT, complete and private. Furthermore, we analyze its complexity.

We first analyze the complexity of TP and prove it is PPT. We denote, as in the protocol, by $k, L, m, |S|$ the number of features, labels, decision tree nodes and considered thresholds, respectively, and by λ the security parameter. First we analyze $\mathsf{Srv}_{\mathsf{TP}}$. $\mathsf{Srv}_{\mathsf{TP}}$ performs in the protocol $n \cdot L$ homomorphic multiplications and n homomorphic additions for each leaf (Step 1, Figure 6) as well as for each internal node and each threshold and feature (Step 2a, Figure 6), plus another k homomorphic multiplications and additions to process the response from $\mathsf{Clnt}_{\mathsf{TP}}$ (Step 2d, Figure 6). Each homomorphic operation is polynomial in the security parameter λ. Therefore, $\mathsf{Srv}_{\mathsf{TP}}$ is PPT and with overall complexity of $O(n \cdot m \cdot |S| \cdot k \cdot L) \cdot poly(\lambda)$. Next we analyze $\mathsf{Clnt}_{\mathsf{TP}}$. $\mathsf{Clnt}_{\mathsf{TP}}$ performs in the computation phase (Figure 5 Step 2) $O(k \cdot |S| \cdot L)$ operations of Dec (Step 2b, Figure 6) and $O(k)$ operations of Enc (Step 2c, Figure 6)

Shared parameters: An FHE encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$; a security parameter λ; a soft-step function ϕ obtained via Equation 2; the numbers of features k and labels L, the set of considered thresholds S, and the depth $\mathsf{maximal\_depth}$ of the decision tree to be constructed.

Client's input: A set of n examples $\mathcal{X}$ and the corresponding labels $\mathcal{Y}$, where each example $x \in \mathcal{X}$ is in $[-1, 1]^k$ and the corresponding $y_x \in \mathcal{Y}$ is a 1-hot encoding of the label of x.

Client's output: A tree $\mathsf{T} = (V, E)$, where each node $v \in V$ is a data structure containing the following fields: $v.feature \in [k]$ and $v.\theta \in S$ denoting the feature and threshold associated with the node, and $v.leaf\_value$ that is a L-dimension vector of values in $\mathbb{R}$ if v is a leaf. In addition, each node has $v.right$ and $v.left$ that denote the right and left children of v, respectively.

Server's input and output: The server has no input and receives no output.

The protocol $\mathsf{TP} = \langle \mathsf{Clnt_{TP}}, \mathsf{Srv_{TP}} \rangle$ proceeds as follows (denoting encrypted values within $[\![\cdot]\!]$):

1. **Input outsourcing phase:**
   (a) $\mathsf{Clnt_{TP}}$ runs $\mathsf{Gen}(1^\lambda)$ to obtain a public and secret key pair (pk, sk).
   (b) $\mathsf{Clnt_{TP}}$ encrypts each example $x \in \mathcal{X}$ and the corresponding label $y_x \in \mathcal{Y}$, entry-by-entry, to obtain vectors $\mathbf{c_x}$ and $\mathbf{c_{y_x}}$ of ciphertexts for x and y_x respectively. $\mathsf{Clnt_{TP}}$ sends the set of ciphertexts $\mathsf{CTXT} = \{\mathbf{c_x}, \mathbf{c_{y_x}}\}_{x \in \mathcal{X}, y_x \in \mathcal{Y}}$ together with pk to $\mathsf{Srv_{TP}}$.
2. **Computation phase:** For each node v we denote by $\mathsf{W}_v = \{[\![w_x]\!]\}_{x \in \mathcal{X}}$ a set of encrypted weights associated with v. We initialize all $[\![w_x]\!] \in \mathsf{W}_{root}$ to $\mathsf{Enc}_{pk}(1)$; and initialize a variable d indicating the currently constructed tree depth to 0. For each depth $d = 1, \ldots, \mathsf{maximal\_depth}$ and for each node v at depth d, $\mathsf{Srv_{TP}}$ constructs v by invoking the sub-protocol $\mathsf{Enc\_Train}$ (Figure 6), executed with $\mathsf{Srv_{TP}}$ holding $(pk, \mathsf{CTXT}, \mathsf{W}_v, d, v)$ and $\mathsf{Clnt_{TP}}$ holding sk, to obtain encrypted values for $v.feature$, $v.\theta$, $\mathsf{W}_{v.right}$ and $\mathsf{W}_{v.left}$ ($v.leaf\_value$ when v is a leaf).
3. **Output phase:** $\mathsf{Srv_{TP}}$ sends to $\mathsf{Clnt_{TP}}$ the trained tree in encrypted form, $\mathsf{Clnt_{TP}}$ decrypts and outputs the cleartext tree.

Fig. 5. The training protocol $\mathsf{TP} = \langle \mathsf{Clnt_{TP}}, \mathsf{Srv_{TP}} \rangle$, constructing a decision tree T in BFS manner. The computation phase (Step 2) is an adaptation of Algorithm 2 to a protocol that operates over encrypted data (denoting encrypted values by placing them within $[\![\cdot]\!]$). The protocol can be executed in parallel to train a Random Forest.

and computes Gini on cleartext. The time to compute each Enc and Dec is polynomial in the security parameter λ, and the time to compute Gini on cleartext is $O(k \cdot |S| \cdot L)$ (Figure 2). So the complexity of $\mathsf{Clnt_{TP}}$ in the computation phase (Figure 5 Step 2) is $O(m \cdot k \cdot |S| \cdot L) \cdot poly(\lambda)$. Moreover, the entire computation of $\mathsf{Clnt_{TP}}$ (including generating keys, encrypting the input and decrypting the output in Figure 5, Steps 1 and Step 3) is polynomial in its input and the security parameter, so $\mathsf{Clnt_{TP}}$ is PPT. Finally we analyze the communication complexity. At each node

Sub-protocol $\mathsf{Enc\_Train}$ executed with $\mathsf{Srv_{TP}}$ holding $(pk, \mathsf{CTXT}, \mathsf{W}_v, d, v)$, $\mathsf{Clnt_{TP}}$ holding sk, and shared parameters as in Figure 5, where $\mathsf{CTXT} = \{\mathbf{c_x}, \mathbf{c_{y_x}}\}_{x\in\mathcal{X}, y_x\in\mathcal{Y}}$ is a set of encrypted examples and labels, $\mathsf{W}_v = \{[\![w_x]\!]\}_{x\in\mathcal{X}}$ is a set of encrypted weights for the node v, and d is the depth of v, where v is the node to be constructed.

The sub-protocol proceeds as follows (denoting encrypted values within $[\![\cdot]\!]$):

1. If d is the $\mathsf{maximal\_depth}$, $\mathsf{Srv_{TP}}$ homomorphically evaluates (using Eval_{pk}):
$$v.leaf\_value = \sum_{x\in\mathcal{X}} [\![w_x]\!] \cdot \mathbf{c_{y_x}}.$$
2. Otherwise, the protocol proceeds as follows:
   (a) For each feature $i \in [k]$ and each threshold $\theta \in S$, $\mathsf{Srv_{TP}}$ homomorphically evaluates (using Eval_{pk}):
$$[\![\mathsf{right}[i,\theta]]\!] \leftarrow \sum_{x\in X} [\![w_x]\!] \cdot \phi\big(\mathbf{c_x}[i] - \theta\big) \cdot \mathbf{c_{y_x}}$$
$$[\![\mathsf{left}[i,\theta]]\!] \leftarrow \sum_{x\in X} [\![w_x]\!] \cdot \phi\big(\theta - \mathbf{c_x}[i]\big) \cdot \mathbf{c_{y_x}}$$
   and sends to $\mathsf{Clnt_{TP}}$ the resulting L-dimensional vectors of ciphertexts:
$$\{[\![\mathsf{right}[i,\theta]]\!], [\![\mathsf{left}[i,\theta]]\!]\}_{i\in[k],\theta\in S}$$
   (b) $\mathsf{Clnt_{TP}}$ decrypts (using $\mathsf{Dec}_{sk}(\cdot)$) to obtain $\mathsf{right}[i,\theta]$ and $\mathsf{left}[i,\theta]$, for all $i \in [k]$ and $\theta \in S$, and computes the Gini impurity (Figure 2) on these cleartext values to obtain the feature i^* and threshold θ^* for the constructed node v:
$$(i^*, \theta^*) \leftarrow \mathsf{Gini}\left(\{\mathsf{right}[i,\theta], \mathsf{left}[i,\theta]\}_{i\in[k],\theta\in S}\right)$$
   (c) $\mathsf{Clnt_{TP}}$ encrypts i^* in 1-hot encoding and θ^* (using $\mathsf{Enc}_{pk}(\cdot)$) and sends the resulting ciphertexts $[\![i^*]\!]$ and $[\![\theta^*]\!]$ to $\mathsf{Srv_{TP}}$. (In case any error occurs during the decryption or calculation, then $\mathsf{Clnt_{TP}}$ encrypts and sends to $\mathsf{Srv_{TP}}$ an encryption of an arbitrary feature and threshold of the appropriate length.)
   (d) $\mathsf{Srv_{TP}}$ sets the encrypted values for $v.feature$ and $v.\theta$ to be $[\![i^*]\!]$ and $[\![\theta^*]\!]$ respectively; for each $x \in \mathcal{X}$, homomorphically evaluates (using Eval_{pk}):
$$[\![w_x^{\mathsf{right}}]\!] \leftarrow [\![w_x]\!] \cdot \phi\Big(\big(\sum_{j\in[k]} \mathbf{c_x}[j] \cdot [\![i^*]\!][j]\big) - [\![\theta^*]\!]\Big)$$
$$[\![w_x^{\mathsf{left}}]\!] \leftarrow [\![w_x]\!] \cdot \phi\Big([\![\theta^*]\!] - \big(\sum_{j\in[k]} \mathbf{c_x}[j] \cdot [\![i^*]\!][j]\big)\Big)$$
   and sets $\mathsf{W}_{v.right} = \{[\![w_x^{\mathsf{right}}]\!]\}_{x\in\mathcal{X}}$ and $\mathsf{W}_{v.left} = \{[\![w_x^{\mathsf{left}}]\!]\}_{x\in\mathcal{X}}$.

Fig. 6. The sub-protocol $\mathsf{Enc\_Train}$ constructs the node v from the encrypted examples, labels and their weights at v. The outcome is encrypted values for $v.feature$, $v.\theta$, $\mathsf{W}_{v.right}$ and $\mathsf{W}_{v.left}$ when v is an internal node ($v.leaf\_value$ when v is a leaf).

$O(k \cdot |S| \cdot L)$ ciphertexts are transmitted, and hence the communication complexity is $O(m \cdot k \cdot |S| \cdot L) \cdot poly(\lambda)$.

Next we prove that $\mathsf{TP} = \langle \mathsf{Clnt_{TP}}, \mathsf{Srv_{TP}} \rangle$ is complete. Observe that $\langle \mathsf{Clnt_{TP}}, \mathsf{Srv_{TP}} \rangle$ homomorphically evaluates the same function as computed in Algorithm 2. So completeness follows immediately from the correctness of $\mathcal{E}$.

Finally, we prove that TP satisfies the privacy condition of Definition 5. Assume by contradiction that privacy does not hold for TP. That is, there exists a PPT distinguisher $\mathcal{D}$ that chooses $(\mathcal{X}_0, \mathcal{Y}_0), (\mathcal{X}_1, \mathcal{Y}_1) \in \mathsf{A}$ with $|\mathcal{X}_0| = |\mathcal{X}_1|$, and $|\mathcal{Y}_0| = |\mathcal{Y}_1|$, and a polynomial $p(\cdot)$ such that for infinitely many $\lambda \in \mathbb{N}$:

$$\begin{aligned} &\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_1, \mathcal{Y}_1), \perp, \lambda)) = 1] \\ &- \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_0, \mathcal{Y}_0), \perp, \lambda)) = 1] \geq p(\lambda) \end{aligned} \tag{15}$$

We show below that given $\mathcal{D}$ we can construct an adversary $\mathcal{A}$ that violate the CPA security of $\mathcal{E}$.

The adversary $\mathcal{A}$ participates in $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}$ as follows:

1. Upon receiving pk output $(\mathcal{X}_0, \mathcal{Y}_0), (\mathcal{X}_1, \mathcal{Y}_1)$.
2. Upon receiving $\mathsf{CTXT} \leftarrow \mathsf{Enc}_{pk}(\mathcal{X}_b, \mathcal{Y}_b)$ behave exactly as $\mathsf{Srv_{TP}}$ behaves while executing TP upon receiving $\mathsf{CTXT} = \{\mathbf{c_x}, \mathbf{c_{y_x}}\}$ and pk from $\mathsf{Clnt_{TP}}$, except that every message $[\![\mathsf{right}[i,\theta]]\!], [\![\mathsf{left}[i,\theta]]\!]_{i \in [k], \theta \in S}$ sent from $\mathsf{Srv_{TP}}$ to $\mathsf{Clnt_{TP}}$ is answered by $\mathcal{A}$ as follows: $\mathcal{A}$ samples uniformly at random $i \leftarrow [k]$ and $\theta \leftarrow S$, computes $\mathsf{Enc}_{pk}(i, \theta)$, and sends this ciphertext to $\mathsf{Srv_{TP}}$ as if it were the response from $\mathsf{Clnt_{TP}}$.
3. Run the distinguisher $\mathcal{D}$ on $\mathsf{view}_{\mathsf{Srv_{TP}}}$ ($\mathsf{Srv_{TP}}$'s view in $\mathcal{A}$ during step 2) and output whatever $\mathcal{D}$ outputs.

The adversary $\mathcal{A}$ is PPT due to $(\mathcal{X}_0, \mathcal{Y}_0), (\mathcal{X}_1, \mathcal{Y}_1)$ being efficiently samplable and $\mathsf{Srv_{TP}}$ and $\mathcal{D}$ being PPT. Note that TP is almost perfectly simulated except that the queries to $\mathsf{Clnt_{TP}}$ are simulated using encryption of randomly sampled elements. Let TP' denote this variant of TP that is simulated by $\mathcal{A}$, namely TP' is a protocol identical to TP except that each query to $\mathsf{Clnt_{TP}}$ is answered by the encryption of a randomly sampled pair $(i, \theta) \leftarrow [k] \times S$. We denote by $\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_b, \mathcal{Y}_b), \perp, \lambda)$ the view of $\mathsf{Srv_{TP}}$, simulated by $\mathcal{A}$, in the execution of $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}$ with bit b being selected by the challenger. By definition of TP' it holds that for every $b \in \{0,1\}$,

$$\begin{aligned} &\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}'}_{\mathsf{Srv_{TP}}}((\mathcal{X}_b, \mathcal{Y}_b), \perp, \lambda)) = 1] \\ &= \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_b, \mathcal{Y}_b), \perp, \lambda)) = 1] \end{aligned} \tag{16}$$

Furthermore, the CPA security of $\mathcal{E}$ guarantees that the server's view in TP and TP' is computationally indistinguishable (see Lemma 2). Putting together Lemma 2 and Equation 15 it follows that

$$\begin{aligned}\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}'}_{\mathsf{Srv_{TP}}}((\mathcal{X}_1,\mathcal{Y}_1),\perp,\lambda))=1]\\ -\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}'}_{\mathsf{Srv_{TP}}}((\mathcal{X}_0,\mathcal{Y}_0),\perp,\lambda))=1]\geq p(\lambda)-\mathsf{neg}(\lambda)\ .\end{aligned} \tag{17}$$

Next, from Equations 16 and 17 it follows that:

$$\begin{aligned}\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_1,\mathcal{Y}_1),\perp,\lambda))=1]\\ -\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_0,\mathcal{Y}_0),\perp,\lambda))=1]\geq p(\lambda)-\mathsf{neg}(\lambda).\end{aligned} \tag{18}$$

Therefore, we obtain that:

$$\begin{aligned}&\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda)=1]\\
&=\frac{1}{2}\cdot\big(\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda)=1|b=1]+\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda)=1|b=0]\big)\\
&=\frac{1}{2}\cdot\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_1,\mathcal{Y}_1),\perp,\lambda))=1]\\
&+\frac{1}{2}\cdot\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_0,\mathcal{Y}_0),\perp,\lambda))=0]\\
&=\frac{1}{2}+\frac{1}{2}\cdot\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_1,\mathcal{Y}_1),\perp,\lambda))=1]\\
&-\frac{1}{2}\cdot\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{EXP}^{cpa}}_{\mathsf{Srv_{TP}}}((\mathcal{X}_0,\mathcal{Y}_0),\perp,\lambda))=1]\\
&\geq\frac{1}{2}+\frac{1}{2}\cdot p(\lambda)-\mathsf{neg}(\lambda)\end{aligned}$$

where the last inequality follows from Equation 18. Combining this with $\mathcal{A}$ being PPT we derive a contradiction to $\mathcal{E}$ being CPA secure. This concludes the proof. □

Let $\mathsf{TP}'=\langle\mathsf{Clnt}'_{\mathsf{TP}},\mathsf{Srv_{TP}}\rangle$ be as defined in the proof of Theorem 3, i.e., is identical to $\mathsf{TP}=\langle\mathsf{Clnt_{TP}},\mathsf{Srv_{TP}}\rangle$ except that $\mathsf{Clnt}'_{\mathsf{TP}}$ samples $(i,\theta)\leftarrow[k]\times S$ at random instead of executing step 2b, Figure 6. We show that the server is indifferent to the correctness of answers it receives from the client in the sense that its view is in TP and TP' is indistinguishable.

Lemma 2. *For every efficiently samplable $(\mathcal{X},\mathcal{Y})\in\mathsf{A}$, and all $\lambda\in\mathbb{N}$ the following holds:*

$$\mathit{view}^{\mathsf{TP}'}_{\mathsf{Srv_{TP}}}((\mathcal{X},\mathcal{Y}),\perp,\lambda)\approx_c \mathit{view}^{\mathsf{TP}}_{\mathsf{Srv_{TP}}}((\mathcal{X},\mathcal{Y}),\perp,\lambda)$$

Proof. Assume by contradiction that Lemma 2 does not hold. That is, there exists a PPT distinguisher $\mathcal{D}$ that chooses $(\mathcal{X}, \mathcal{Y}) \in \mathsf{A}$ and a polynomial $p(\cdot)$ such that for infinitely many $\lambda \in \mathbb{N}$:

$$\begin{aligned} &\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}'}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)) = 1] \\ &- \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{TP}}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)) = 1] \geq p(\lambda) \ . \end{aligned} \tag{19}$$

We define a series of hybrid executions that gradually move between $\mathsf{TP} = \langle \mathsf{Clnt_{TP}}, \mathsf{Srv_{TP}} \rangle$ execution (where Gini impurity is used) to $\mathsf{TP}' = \langle \mathsf{Clnt}'_{\mathsf{TP}}, \mathsf{Srv_{TP}} \rangle$ execution (where random i and θ are used). Let q denote the number of queries made to $\mathsf{Clnt_{TP}}$ in Figure 5 ($\mathsf{Srv_{TP}}$ makes a single query to $\mathsf{Clnt_{TP}}$ per constructed node in T). We define $q + 1$ hybrids as follows:

Hybrid H_0 is defined as the execution of $\langle \mathsf{Clnt_{TP}}, \mathsf{Srv_{TP}} \rangle$.

Hybrid H_j $(j = 1, \ldots, q)$ is similar to H_0 except that the last j queries to $\mathsf{Clnt_{TP}}$ are answered by sampling uniformly random $(i, \theta) \leftarrow [k] \times S$ and responding with $\mathsf{Enc}_{pk}(i, \theta)$ (instead of executing 2b in Figure 6).

Note that in each pair of adjacent hybrids H_{j-1} and H_j for $j \in [q]$ the difference is that in H_j the $(q + 1 - j)$'th query is answered using random (i, θ) instead of those maximizing the Gini impurity. Denote by $\mathsf{view}^{\mathsf{H}_j}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)$ the view of $\mathsf{Srv_{TP}}$ in the hybrid H_j.

By the hybrid argument it follows from Equation 19 there exists $j \in [q]$ such that:

$$\begin{aligned} &\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{H}_j}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)) = 1] \\ &- \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{H}_{j-1}}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)) = 1] \geq \frac{p(\lambda)}{q} \end{aligned} \tag{20}$$

We show that Equation 20 contradicts $\mathcal{E}$ being CPA secure. That is, we construct an adversary $\mathcal{A}$ that communicates with the challenger Chal in the CPA indistinguishability experiment $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}$ and forces the output to be 1 with a non-negligible advantage over half. Concretely, $\mathcal{A}$ participates in $\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}$ as follows:

1. $\mathcal{A}$ executes Algorithm 2 on $(\mathcal{X}, \mathcal{Y})$ to compute for each $v \in \mathsf{T}$, the associated feature $v.feature$ and threshold $v.\theta$.
2. Upon receiving pk from Chal, $\mathcal{A}$ computes $\mathsf{CTXT} \leftarrow \mathsf{Enc}_{pk}(\mathcal{X}, \mathcal{Y})$ and executes $\mathsf{Srv_{TP}}$ on (CTXT, pk) while answering each query that $\mathsf{Srv_{TP}}$ as follows:

(a) For the $q-j$ first queries of $\mathsf{Srv_{TP}}$, $\mathcal{A}$ encrypts under pk the feature $v.feature$ and threshold $v.\theta$ associated with the queried node v, and sends the resulting ciphertexts to $\mathsf{Srv_{TP}}$.

(b) For the $(q-j+1)$'th query of $\mathsf{Srv_{TP}}$, $\mathcal{A}$ proceeds as follows:

   i. Denote by (i_0, θ_0) the feature and threshold associated with the queried node. $\mathcal{A}$ samples uniformly random $(i_1, \theta_1) \leftarrow [k] \times S$, and sends (i_0, θ_0) and (i_1, θ_1) to Chal.

   ii. Upon receiving from Chal the challenge ciphertext $c \leftarrow Enc_{pk}(i_b, \theta_b)$ for uniformly random $b \leftarrow \{0,1\}$, $\mathcal{A}$ forwards this ciphertext c to $\mathsf{Srv_{TP}}$.

(c) For the rest of the queries, $\mathcal{A}$ samples uniformly random $(i, \theta) \leftarrow [k] \times S$, and sends $\mathsf{Enc}_{pk}(i, \theta)$ to $\mathsf{Srv_{TP}}$.

3. $\mathcal{A}$ executes the distinguisher $\mathcal{D}$ on the view of $\mathsf{Srv_{TP}}$ during the execution of Step 2 above, denoted $\mathsf{views}_{\mathsf{Srv_{TP}}}$, and outputs whatever $\mathcal{D}$ outputs.

We note that if $b = 0$, then the challenge ciphertext c is the encryption of $(v.feaure, v.\theta)$ and $\mathsf{views}_{\mathsf{Srv_{TP}}}$ is exactly as in H_{j-1} and otherwise as in H_j. Therefore, we obtain that

$$\begin{aligned}
&\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1] \\
=&\frac{1}{2} \cdot \big(\Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1 | b = 1] + \Pr[\mathsf{EXP}^{cpa}_{\mathcal{A},\mathcal{E}}(\lambda) = 1 | b = 0] \big) \\
=&\frac{1}{2} \cdot (\Pr[\mathcal{D}(\mathsf{view}^{\mathsf{H}_j}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)) = 1] \\
&\quad 1 - \Pr[\mathcal{D}(\mathsf{view}^{\mathsf{H}_{j-1}}_{\mathsf{Srv_{TP}}}((\mathcal{X}, \mathcal{Y}), \perp, \lambda)) = 1]) \geq \frac{1}{2} + \frac{1}{2} \cdot \frac{p(\lambda)}{q}
\end{aligned} \tag{21}$$

which concludes the proof. □

5 Implementation Details and Experimental Results

We empirically evaluated our decision trees algorithms and protocols for both accuracy and run-time performance on encrypted data (see Sections 5.1-5.2, respectively). Our evaluations were done with respect to a single decision tree, and can naturally be extended to random forests where trees are trained/evaluated in parallel. The soft-step function we employed is a polynomial of degree 15, constructed via Equation 2 with a weighting function $w\colon [-2, 2] \to [0, 1]$ defined to be zero in the interval

$[-0.2, 0.2]$ and a constant positive value elsewhere. For training, we used thresholds on a 0.05 grid in the $[-1, 1]$ interval. We used standard UCI repository datasets [13] in our evaluation, ranging in size from very small (*iris* with 4 features and 150 examples) to the moderate size common in real-world applications (*forest cover* with 54 features and over half a million examples).

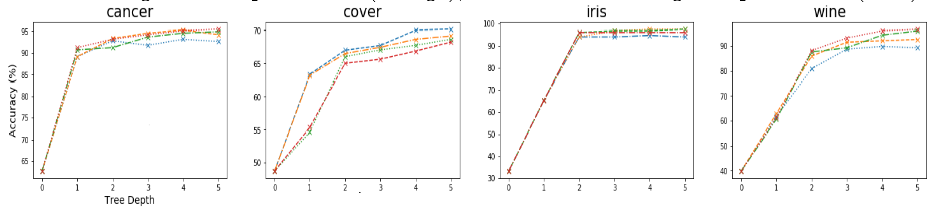
5.1 Accuracy of our Decision-Tree Algorithms

The accuracy of our Algorithms 1-2 was evaluated in comparison to standard trees, on the benchmark datasets. We used a 3-fold cross-validation procedure, where each dataset was randomly partitioned into three equal-size parts, and each of the three parts serves as a test-set for a classifier trained on the remaining two. The *overall accuracy* is calculated as the percentage of correct classification on test examples (each example in the data is taken exactly once, so the accuracy reported is simply the percentage of examples that were correctly classified).

We compared all four possible combinations of training and prediction according to our algorithms vs. the standard algorithms; see Figure 7. We trained trees up to depth 5, as customary when using random forests.

The results show an overall comparable accuracy, indicating that our Algorithms 1-2 are a valid replacement for standard decision trees in terms of accuracy.

Fig. 7. Accuracy of ours vs. scikit-learn [30] algorithms on four UCI datasets and tree depth 0–5 (depth 0 is the majority-class baseline), in four execution modes: our training and prediction (red), our training and scikit-learn prediction (green), scikit-learn training and our prediction (orange), scikit-learn training and prediction (blue).



5.2 Running-Time on Encrypted Data

We implemented our training and prediction protocols over data encrypted with CKKS homomorphic encryption scheme [9] in Microsoft

SEAL v3.2.2 [36]. We set SEAL parameters to security-level 80 bits and irreducible polynomial degree 8192. We ran experiments with depth 4 trees.

Training over encrypted datasets. Our experiments were executed on AWS x1.16xlarge as the server. The input examples are encrypted feature-by-feature, while packing 4096 values in each ciphertext; the associated labels (in 1-hot encoding) are likewise encrypted in a packed form.

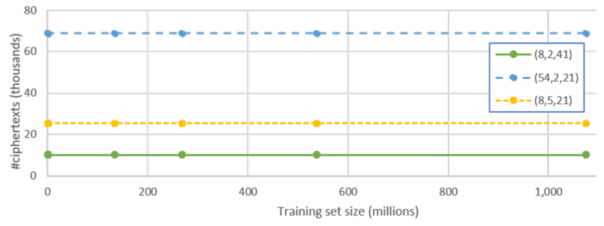
The server run-time on encrypted UCI datasets (subs-sampled) ranged from minutes on small datasets to hours on medium ones; see Table 1. Executions on encrypted synthetic datasets containing up to $131,000$ encrypted samples and (features,labels,splits)=$(8, 2, 21)$ exhibited a server run-time of under a day.

dataset name	# training examples	# features	# labels	Server training time (minutes)
iris	100	4	3	19
wine	119	13	3	59
cancer	381	30	2	107
digits	1,203	64	10	665
cover	10,000	54	7	1220

Table 1. Server run-time for training depth 4 decision trees on encrypted UCI datasets

The client (*e.g.*, KMS) run-time in all experiments ranged from seconds to under three minutes. The communication complexity is independent of the dataset size. See Figure 8 for the number of transmitted ciphertexts. The ciphertext size is roughly 0.25MB after compression.

Fig. 8. Total number of transmitted ciphertexts vs. the training set size, when training a full tree of depth 4 on various (features,labels,splits) settings



Prediction over encrypted data was performed on encrypted examples and cleartext trees, and executed on a personal Intel Core i7-4790 CPU

16GB memory computer (using a single core). The average run-time was 1.15 second.

6 Conclusions

In this work we present FHE-friendly algorithms and protocols for decision tree based prediction and training over encrypted data that are the first to attain all the following desired properties: non-interactive prediction, lightweight client and communication in training, and rigorous privacy guarantee. We ran extensive experiments on standard UCI and synthetic datasets, all encrypted with fully homomorphic encryption, demonstrating high accuracy comparable to standard algorithms on cleartext data, fast prediction (1.15 seconds), and feasible training (minutes to hours). Our protocols support real-life enterprise use-cases, and are well suited for offline execution, *e.g.* in nightly batched prediction.

As a central technique we devised a soft-step function yielding a low-degree weighted approximation for the step function. This may have further applications, beyond tree-based models, including *neural networks* based prediction utilizing our soft-step function to replace *sigmoid* or *sign* activation to gain speedup.

References

1. M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider. Secure evaluation of private linear branching programs with medical applications. In *European Symposium on Research in Computer Security*, pages 424–439. Springer, 2009.
2. M. Blatt, A. Gusev, Y. Polyakov, K. Rohloff, and V. Vaikuntanathan. Optimized homomorphic encryption solution for secure genome-wide association studies. Cryptology ePrint Archive, Report 2019/223, 2019. `https://eprint.iacr.org/2019/223`.
3. R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.
4. Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 868–886, 2012.
5. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325, 2012.
6. J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel. Privacy-preserving remote diagnostics. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 498–507. ACM, 2007.

7. H. Chen, R. Gilad-Bachrach, K. Han, Z. Huang, A. Jalali, K. Laine, and K. Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC medical genomics*, 11(4):81, 2018.
8. H. Chen, R. Gilad-Bachrach, H. Kyoohyung, Z. Huang, A. Jalali, K. Laine, and K. Lauter. Logistic regression over encrypted data from fully homomorphic encryption. *BMC Medical Genomics*, 11, 10 2018.
9. J. H. Cheon, A. Kim, M. Kim, and Y. S. Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pages 409–437, 2017.
10. M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. Nascimento, W.-S. Poon, and S. Truex. Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. *IEEE Transactions on Dependable and Secure Computing*, 16(2):217–230, 2017.
11. S. de Hoogh, B. Schoenmakers, P. Chen, and H. op den Akker. Practical secure decision tree learning in a teletreatment application. In *International Conference on Financial Cryptography and Data Security*, pages 179–194. Springer, 2014.
12. W. Du and Z. Zhan. Building decision tree classifier on private data. In *Proceedings of the IEEE international conference on Privacy, security and data mining-Volume 14*, pages 1–8. Australian Computer Society, Inc., 2002.
13. D. Dua and C. Graff. UCI machine learning repository, 2017.
14. F. Emekci, O. D. Sahin, D. Agrawal, and A. El Abbadi. Privacy preserving decision tree learning over multiple parties. *Data Knowl. Eng.*, 63(2):348–361, Nov 2007.
15. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
16. C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.
17. R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.
18. C. Hazay and Y. Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.
19. E. Hesamifard, H. Takabi, M. Ghasemi, and R. Wright. Privacy-preserving machine learning as a service. *Proceedings on Privacy Enhancing Technologies*, 2018:123–142, 06 2018.
20. M. Joye and F. Salehi. Private yet efficient decision tree evaluation. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 243–259. Springer, 2018.
21. A. Kim, Y. Song, M. Kim, K. Lee, and J. Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC Medical Genomics*, 11, 10 2018.
22. A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11(4):83, 2018.
23. M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR Medical Informatics*, 6, 08 2017.

24. Á. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider. Sok: Modular and efficient private decision tree evaluation. *PoPETs*, 2019(2):187–208, 2019.
25. H. Kyoohyung, S. Hong, J. Cheon, and D. Park. Logistic regression on homomorphic encrypted data at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9466–9471, 07 2019.
26. H. Laurent and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.
27. Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54. Springer, 2000.
28. P. Lory. Enhancing the efficiency in privacy preserving learning of decision trees in partitioned databases. In J. Domingo-Ferrer and I. Tinnirello, editors, *Privacy in Statistical Databases*, pages 322–335, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
29. K. Nandakumar, N. K. Ratha, S. Pankanti, and S. Halevi. Towards deep neural network training on encrypted data. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, page 0. Computer Vision Foundation / IEEE, 2019.
30. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
31. J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
32. E. Y. Remez. Sur la détermination des polynômes d'approximation de degré donnée. *Comm. Soc. Math. Kharkov*, 10(4163):196, 1934.
33. R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
34. T. J. Rivlin. *An introduction to the approximation of functions*. Courier Corporation, 2003.
35. S. Samet and A. Miri. Privacy preserving id3 using gini index over horizontally partitioned data. In *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, AICCSA '08, pages 645–651, Washington, DC, USA, 2008. IEEE Computer Society.
36. Microsoft SEAL (release 3.3). `https://github.com/Microsoft/SEAL`, 2019. Microsoft Research, Redmond, WA.
37. R. K. Tai, J. P. Ma, Y. Zhao, and S. S. Chow. Privacy-preserving decision trees evaluation via linear functions. In *European Symposium on Research in Computer Security*, pages 494–512. Springer, 2017.
38. A. Tueno, F. Kerschbaum, and S. Katzenbeisser. Private evaluation of decision trees using sublinear cost. *PoPETs*, 2019(1):266–286, 2019.
39. J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson. Privacy-preserving decision trees over vertically partitioned data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(3):14, 2008.
40. K. Wang, Y. Xu, R. She, and P. S. Yu. Classification spanning private databases. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 293–298. AAAI Press, 2006.
41. D. J. Wu, T. Feng, M. Naehrig, and K. Lauter. Privately evaluating decision trees and random forests. *Proceedings on Privacy Enhancing Technologies*, 2016(4):335–355, 2016.

42. M.-J. Xiao, L.-S. Huang, Y.-L. Luo, and H. Shen. Privacy preserving id3 algorithm over horizontally partitioned data. In *Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, PDCAT '05, pages 239–243, Washington, DC, USA, 2005. IEEE Computer Society.