Secure Training of Random Forest Classifiers over Continuous Data

Jianwei Shen

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Computer Science and Systems

University of Washington

2020

Reading Committee:

Dr. Martine De Cock, Chair

Dr. Anderson Nascimento

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington

Abstract

Secure Training of Random Forest Classifiers over Continuous Data

Jianwei Shen

Chair of the Supervisory Committee:
Professor Dr. Martine De Cock
School of Engineering and Technology

Existing Secure Multi-Party Computation (MPC) protocols for privacy-preserving training of decision trees over distributed data assume that the attributes are categorical. In real-life applications, attributes are often numerical. The standard "in the clear" algorithm to grow decision trees on data with continuous values requires sorting of training examples for each attribute in each node in the quest for an optimal cut-point in the range of attribute values. Sorting is a prohibitively expensive operation in MPC, hence secure protocols that mimic the traditional decision tree training algorithm are very inefficient. In this thesis we propose an alternative, more efficient strategy for secure training of decision tree based models on data with continuous attributes, namely secure discretization of the data, followed by secure training of a random forest classifier over the discretized data. In addition to mathematically proving that the proposed approach is correct and secure, we experimentally evaluate it in terms of classification accuracy and runtime on a variety of benchmark data sets. To the best of our knowledge, our approach is the very first to privately train decision tree based models with continuous attributes where the overall complexity depends only linearly on the size of the entire training data set – contrary to existing sorting-based solutions.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I wish to express sincere appreciation to Dr. Anderson C.A. Nascimento and Dr. Martine De Cock, who have been advising me and offering generous help since my first day entering the PPML group at the University of Washington. Other thanks go to the students and co-workers in the group, especially Mr. Davis Railsback, for the fruitful research collaboration throughout my thesis, and for providing clear descriptions and implementations of some of the building blocks (cryptographic subprotocols) that became part of this thesis, in particular the new discretization protocol and its subprotocols. Additionally, I wish to thank Dr. Rafael Dowsley from Bar-Ilan University for the proof of correctness of the cryptographic protocols that make up the algorithm. Also, without the previous work of Dr. Sebastiaan de Hoogh, Dr. Berry Schoenmakers, Dr. Ping Chen, and Dr. Harm op den Akker, the thesis would be facing a more significant challenge of starting from scratch.

"If I have seen further
it is by standing on the shoulders of Giants."

DEDICATION

to my parents and dear friends

to those who are fighting against the COVID-19

Chapter 1

INTRODUCTION

The rapidly developing field of machine learning has resulted in many beneficial applications that make life more convenient, and that can even save lives. Machine learning is highly driven by data. People who want to use machine learning applications are currently often expected to trust the service providers and disclose their personal data for a better service, which puts people's privacy in jeopardy. Recent scandals about data leakage and data abuse are harming the trust between service providers and users. Laws and regulations are enacted to protect people's personal data, such as the GDPR, short for the General Data Protection Regulation, on 25th May 2018, the CCPA, short for the California Consumer Privacy Act, on 1st Jan 2020, etc. On the one hand, these laws are intended to protect people's privacy; on the other hand, they are challenging researchers and service providers to compromise and find a balance between service quality and confidentiality.

Under these circumstances, privacy-preserving machine learning (PPML) has gained its spotlight. PPML solutions draw on a variety of different techniques, such as Homomorphic Encryption (HE), Differential Privacy (DP), and Secure Multiparty Computation (MPC). Among those methodologies, we turn to MPC in this thesis. Compared with HE, MPC is often more efficient, and it allows to obtain results with similar accuracy as in the clear, i.e. on unencrypted data, unlike DP.

Privacy-preserving supervised learning can be divided in two tasks: (1) secure training of a machine learning model over data that is distributed across multiple parties; and (2) secure inference or prediction with an already trained model. In this thesis we are concerned with the most challenging of the tasks, namely training. Recent advances in MPC-based protocols for *training* of machine learning models over distributed data are primarily focused on secure

training of neural network architectures [25, 35, 8, 13, 22]. While deep learning is state-of-the-art for tasks that relate to perception, such as computer vision and natural language processing, in domains with structured information, the best results are often obtained with tree ensemble methods, such as random forests and boosted decision trees [15]. The latter also have the advantages of being faster to train and being easier to interpret.

Advances on MPC-based training of tree based classifiers are fairly limited. While there is work on *secure inference* with pre-trained tree ensembles [21], work on *secure training* itself is limited to the training of individual decision trees (DTs). Several authors have proposed a secure version of Quinlan's ID3 algorithm [26] for training DTs with *categorical* attributes (features) [23, 34, 30, 14]. Existing proposals for training of DTs with *continuous* features [36, 32, 3] are based on Quinlan's C4.5 algorithm [27], an algorithm that involves sorting, which is a time-consuming operation in the MPC setting.

We propose a new strategy for secure training of tree based models over data with continuous attribute values. Our approach does not require sorting of attribute values, allowing us to complete the training process in a much shorter time compared to any of the existing secure DT training methods to date. Our technique relies on privacy-preserving discretization of the range of attribute values.

After discretization, we apply a novel protocol π_{RF} – which we propose in this thesis – for secure training of a random forest (RF) of DTs over data with categorical values. To this end, we extend de Hoogh et al.'s secure DT training algorithm [14] with a subprotocol for privacy-preserving random feature selection, and a subprotocol for privacy-preserving sampling of training instances. Combined with the secure discretization protocol π_{DISC}, this allows us to securely train RFs over data with continuous values. To the best of our knowledge, our protocol π_{RF} is the first MPC-based protocol for privacy-preserving training of a random forest with data distributed across multiple parties.

The remainder of this thesis is structured as follows. In Chapter 2, we recall concepts and algorithms related to training machine learning models in the clear, i.e. in an unencrypted manner, including the ID3 algorithm for training a DT [26] and the algorithm for training

a random forest [5]. Next, in Chapter 3, we provide a brief introduction to MPC, with emphasis on the MPC primitives (cryptographic subprotocols) that we use in Chapter 4. In Chapter 4, we present our new protocol π_{RF} for training a random forest securely with training data that is distributed across multiple parties who can not, or do not want to, disclose this data to each other. Among other things, we explain how we do bagging and subspace sampling – both crucial parts of random forest training – in a privacy-preserving manner. In Chapter 5, we present details about the implementation of π_{RF} in Lynx,[1] an open source framework for PPML. Furthermore, we present an empirical study with running time analyses gained from secure training of random forest classifiers on various benchmark data sets with continuous feature values. We conclude the thesis and present directions for future work in Chapter 6.

The results from this work will be submitted for publication as part of a paper:

> Secure Training of Tree based Models over Continuous Data
> Samuel Adams, Chaitali Choudhary, Martine De Cock, Rafael Dowsley, David Melanson, Anderson Nascimento, Davis Railsback, and **Jianwei Shen**
> In preparation, 2020

[1] https://bitbucket.org/uwtppml/lynx

Chapter 2

RANDOM FOREST IN THE CLEAR

2.1 Random Forest

Random forest is an ensemble algorithm, which is well-known for its "bagging" idea and random feature selection technique [5]. Let S be a training set which contains n instances (each with f features and one target class with c possible different labels). For a random forest model with m trees, the algorithm will sample the training set with replacement s times independently before training each tree. To inject more randomness, the algorithm also randomly selects k features to be used in each tree. The algorithm then trains m shallow trees in parallel using these data sets. The random forest training algorithm is recalled in Algorithm 1. The algorithm ID3 mentioned on Line 4 is explained in Section 2.2.

2.1.1 Bagging and Random Feature Selection

"Bagging" stands for "bootstrap aggregating", which is used to improve the stability and accuracy of machine learning algorithms. Usually, it is applied to decision tree-based algorithms, like the random forest. In the context of the random forest training algorithm, bagging helps reduce variance. Combined with the shallow decision trees of the random forest, it helps the random forest achieve low bias and low variance. Formally, given a training set S of size n, uniformly sampling the set with replacement for s times, the probability that one instance is not in the final set S' is

$$p = (1 - \frac{1}{n})^s \tag{2.1}$$

In the special case when $s = n$, the probability is

$$p = (1 - \frac{1}{n})^n \approx \frac{1}{e} \approx 0.368 \tag{2.2}$$

Algorithm 1: Algorithm for Training a Random Forest Classifier.

Input : A set S with n training samples (each sample has f features and one target class with c possible classifications), the number m of trees in an ensemble, the number of features k used in each tree, the number of samples s used in each tree.

Output: An ensemble of trees $T = t_1, t_2, \ldots, t_m$

for $j \leftarrow 1$ **to** m **do**
    Randomly select k of the f features of S.
    Randomly select with replacement s samples $S' = i_1, i_2, \ldots, i_s$ among all samples of S after features are selected.
    Train a decision tree t_i on S' using ID3.
end
return $T = t_1, t_2, \ldots, t_m$

Thus, the probability for an instance to be sampled at least once is 0.632. This number assures that with a high probability every instance would be sampled and used in a random forest with multiple decision trees, while also promoting diversity among different data sets that are created from the original data set with bagging. Indeed, each such data set would contain only 63.2% of the examples of the original data set. Since the decision tree learning algorithm (see Section 2.2) is an unstable algorithm, diversity in the data sets will lead to diversity in the trees trained on those data sets, making the ensemble of trees (the random forest) stronger as a whole.

In Chapter 4, we emulate the process of sampling of training examples through performing matrix multiplication. To achieve this, one can generate a matrix M_s representing the

sampling. For example, given a data matrix S with $n = 4$ instances and $f = 3$ features,

$$S = \begin{pmatrix} 2 & 1 & 1 \\ 3 & 3 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

selecting the 1st instance twice, and the 3rd and 4th instance once, can be achieved by multiplying

$$M_s = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

with S, to obtain

$$S' = M_s \times S = \begin{pmatrix} 2 & 1 & 1 \\ 2 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \tag{2.3}$$

The idea of instance bagging is extended to "feature bagging", which we call random feature selection in this work. The difference with instance bagging is that random feature selection is done without replacement, i.e. each feature is selected at most once. This process can also be done by a matrix multiplication in practice. e.g., by multiplying

$$S = \begin{pmatrix} 2 & 1 & 1 \\ 3 & 3 & 2 \\ 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$$

with the feature selection matrix

$$M_{fs} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

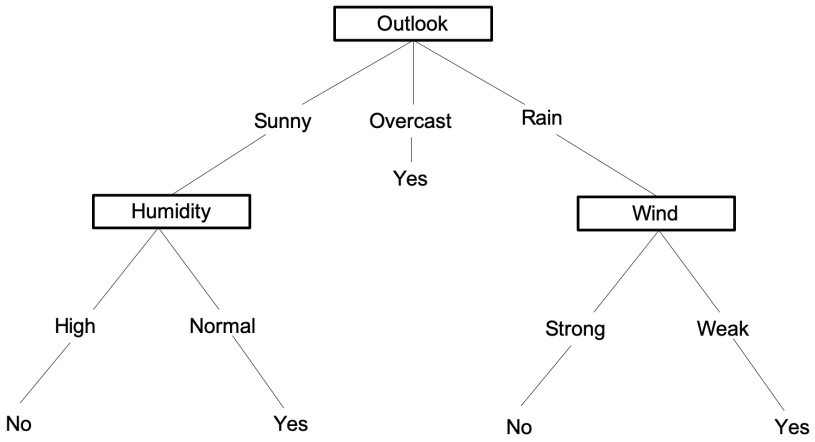

Figure 2.1: Decision tree representation [24]

to obtain

$$S_{fs} = S \times M_{fs} = \begin{pmatrix} 2 & 1 \\ 3 & 3 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} \tag{2.4}$$

in which only the 1st and the 2nd feature are retained.

Typically, for a classification problem with f features, $k = \sqrt{f}$ features are selected.

2.2 Decision Tree and ID3 Algorithm

The decision tree is a highly explainable machine learning model. Decision tree learning can be applied to both classification and regression problems. For a classification problem, the representation of a decision tree is composed of the following contents: 1) each internal node tests an attribute; 2) each branch corresponds to an attribute value; 3) each leaf node contains a class label. An example of a decision tree is given in Figure 2.1.

One of the most crucial things in the decision tree classification problem is to find which attribute should be picked in each internal node to split the data set according to its values. Two criteria are usually adopted, the entropy and gini impurity. Assume that we have a

data set S with f categorical attributes and a class attribute with c different values, i.e. there are c different possible class labels. The set of instances that have value v for attribute a is given by:

$$S_a(v) = \{i \in S | i_a = v\} \tag{2.5}$$

The entropy of $S_a(v)$ with respect to the class attribute is

$$H(S_a(v)) = \sum_{i}^{c} -p_i \log_2 p_i \tag{2.6}$$

So the conditional entropy of S given attribute a is

$$H(S|a) = \sum_{v \in vals(a)} \frac{|S_a(v)|}{|S|} \cdot H(S_a(v)) \tag{2.7}$$

In this work, we adopt the gini impurity instead because it is a huge cost to compute the logarithm in our secure context.

$$I_G(p) = \sum_{i=1}^{c} p_i \sum_{k \neq i} p_k = 1 - \sum_{i=1}^{c} p_i^2 \tag{2.8}$$

The ID3 algorithm [26] using the gini impurity as a split criteria is recalled in Algorithm 2.

Algorithm 2: ID3 Algorithm for Training a Decision Tree Classifier.

Input : A set S with n training samples, target attribute C with c possible classifications, attributes A

Output: A decision tree root R

Create a Root R node for the tree
if *C has the same classification i* **then**
    Assign i to R
    return R
else if *Attributes A is empty,* **then**
    find the most common classification i
    Assign i to R
    return R
else
    find the best attribute a that classifies samples S by gini impurity
    Assign a to R
    for *each possible value v_i of a* **do**
        Add a new tree branch below R, corresponding to the test $a = v_i$
        Let $S_{v_{ai}}$ be the subset of S with the value v_i for a
        if *$S_{v_{ai}}$ is empty* **then**
            find the most common classification i
            Assign i to R
            return R
        else
            ID3($S_{v_{ai}}$,$C_{v_{ai}}$,$A\backslash\{a\}$)
    end
return R

Chapter 3

SECURE MULTIPARTY COMPUTATION

Secure multiparty computation (SMC/MPC) [19] is a subfield of cryptography, which aims at creating procedures to help parties jointly compute the output of a function on combined inputs without each party having to expose their private input. Unlike traditional cryptographic methods that protect parties from adversaries outside the system, SMC focuses on the corruption of parties inside the system. There are two types of corruption in SMC, semi-honest parties and malicious parties. The former ones would be curious about others' secrets but would follow the rules and send out messages correctly, while the latter ones would transmit erroneous information at their own wills. Parties in our context are all semi-honest parties.

3.1 Primitives

Additive secret sharing. There are several secret sharing schemes, i.e., additive secret sharing, Shamir's scheme [31], Blakley's scheme [4], etc. We adopt additive secret sharing schemes. In this scheme, we typically choose a large integer p, and we assume that all inputs belong to the set $\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$ or can be converted to elements of $\mathbb{Z}_p$ (see below for conversion of real valued numbers to integers). $\mathbb{Z}_p$ with addition and multiplication modulo p define a ring. If p is a prime number, $\mathbb{Z}_p$ with addition and multiplication modulo p define a field. In this thesis, we will have protocols that work over a field and a ring. A secret $s \in \mathbb{Z}_p$ can be shared among a number of m parties by giving a random element from $\mathbb{Z}_p$ to each of the parties subject to the restriction these elements add up to the secret modulo p. Note that we must have $p > m$. For ease of explanation we will from now on focus on the case of $n = 2$ and refer to the computing parties as *Alice* and *Bob*. Fig. 3.1 illustrates

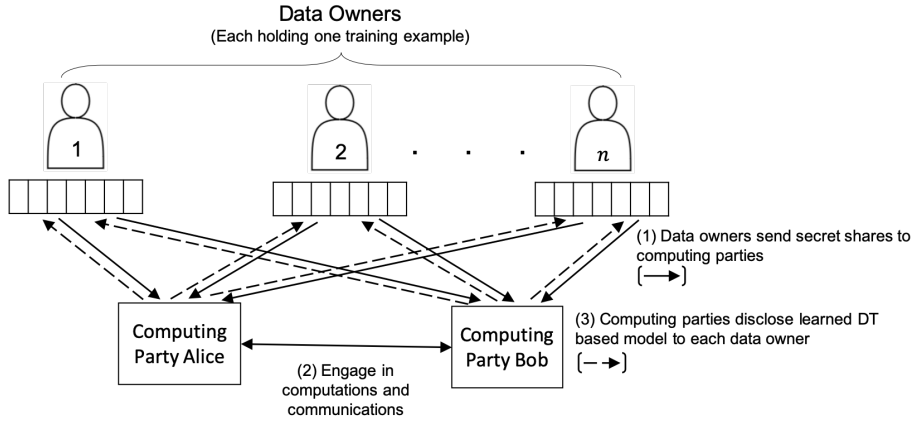

Figure 3.1: Overview of MPC based secure decision tree (DT) based model training. Each of n data owners secret shares their own training data between two computing parties. The computing parties engage in computations and communications to train a DT based model, which is at the end revealed to the data owners. As an alternative to Step (3), our protocols also allow for the learned model to remain hidden, i.e. secret shared between the computing parties. In this case, when a new instance has to be classified, the new instance is secret shared across the computing parties also, who then work together to each obtain secret shares of the class label.

this scenario where n data owners secret share their data with $m = 2$ computing parties. It should be noted that all protocols proposed in this paper can be extended to work for $m > 2$ computing parties with adequate substitutions for the 2PC-exclusive subprotocols π_{trunc}, π_{eq} and π_{geq}.

Formally, given a secret x and n parties,

$$x = \sum_{i=1}^{n} x_i \mod p \tag{3.1}$$

In the special case when all the inputs are binary, as in zeroes and ones, we set p as 2. In this case the secret scheme becomes

$$x = XOR_{i=1}^{n} x_i \tag{3.2}$$

Converting to fixed-point representation. The protocols proposed in this thesis are designed for training of decision tree based models on data with continuous feature values. During the execution of the protocols however, operations are performed on additive shares in a ring $\mathbb{Z}_q$, for some appropriately chosen integer q and two operations, addition and multiplication on it. In this work we mostly use $q = 2^\lambda$, where λ is a positive integer.The feature values x in $\mathbb{R}$ first need to be converted into values $Q(x)$ in $\mathbb{Z}_{2^\lambda}$. To this end we use a fixed point representation with two's complement for negative numbers:

$$Q(x) = \begin{cases} 2^\lambda - \lfloor 2^a \cdot |x| \rfloor & \text{if } x < 0 \\ \lfloor 2^a \cdot x \rfloor & \text{if } x \geq 0 \end{cases} \tag{3.3}$$

When converting $Q(x)$ into its bit representation, it consists of λ bits in total. The first a bits from the right hold the fractional part of x, the next b bits represent the non-negative integer part of x, and the most significant bit (MSB) represents the sign (positive or negative). It is important to choose λ large enough to be able to represent the largest numbers produced during the protocols. When multiplying fixed point numbers, the number of fractional bits doubles and must be truncated to remain in the proper range. Therefore, λ should be chosen to be at least $2(a+b)$. It is also important to choose b that is large enough to represent the maximum possible value of the integer part of all x's.

Arithmetic operations on secret shared values. Given secret shared values $[\![x]\!]_q$ and $[\![y]\!]_q$, and a constant c, Alice and Bob can trivially perform the following operations locally,i.e., without exchanging messages:

- Addition ($z = x + y$): Alice and Bob just add their local shares of x and y. This operation will be denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q + [\![y]\!]_q$.

- Subtraction ($z = x - y$): Alice and Bob subtract their local shares of y from that of x. This operation will be denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q - [\![y]\!]_q$.

- Multiplication by a constant ($z = cx$): Alice and Bob multiply their local shares of x

by c. This operation will be denoted by $[\![z]\!]_q \leftarrow c[\![x]\!]_q$

- Addition of a constant ($z = x + c$): Alice adds c to her share x, while Bob keeps the same share of x. This operation will be denoted by $[\![z]\!]_q \leftarrow [\![x]\!]_q + c$.

Multiplication cannot be performed directly over shares. Additional assumptions are needed. In our case, we achieve multiplication of secrets over shares by assuming a trusted initializer (TI) that pre-distributes correlated randomness to the parties participating in the protocol. This *multiplication triplets* technique was originally proposed by Beaver [2] and is regularly used to enable very efficient solutions in the context of PPML (see e.g. [6, 11, 25, 29, 22]). The TI is not involved in any other part of the execution and does not learn any data from the parties. In case a trusted initializer is not available or desirable, Alice and Bob can simulate the role of the TI with Oblivious Transfer based correlated randomness generation [25] or another available method, at the cost of additional pre-processing time. We use the same protocol π_{DMM} for secure (matrix) multiplication of secret shared values as in [9, 16] and denote by π_{DM} the protocol for the special case of multiplication of scalars and π_{IP} for the inner product.

Truncation. When working with fixed-point representations over $\mathbb{Z}_q$ with a fractional bits, every multiplication generates an extra a bits of unwanted fractional representation. Having a secure way to "chop off" the extra fractional bits generated by multiplication is a requirement to work with fixed-point secret shares. In the two-party scenario, it is possible to perform this truncation with a local probabilistic protocol – hereafter, π_{trunc}– that with overwhelming probability in the security parameter introduces an error of at most 1 [25]. Since all operations to compute π_{trunc} are local, the performance overhead of truncating all multiplication results with this method is essentially zero.

Semi-honest. In this work,we will consider a more general solution to multiparty computation, where we remove the first two restrictions: we will consider any number $n \geq 3$ of players, and we will be able to show that as long as at most $t < n/2$ of the players go together after the protocol is executed and pool all their information, they will learn nothing

more than their own inputs and the outputs they were supposed to receive, even if their computing power is unbounded. We will still assume, however, that all players follow the protocol. This is known as semi-honest or passive security.

Chapter 4

SECURE RANDOM FOREST CLASSIFIER

As is known, to train a random forest, we have to set the number of trees in the forest. When training each tree, we can break the standard random forest training into three steps, which are random feature selection, bootstrap sampling, and training a decision tree using the ID3 algorithm. In the settings of our work, we rely on the SID3 protocol [14] as the underlying protocol to train a decision tree securely. This protocol is derived from the ID3 algorithm, which means that it can only deal with discrete data. Our strategy for handling continuous data in this thesis is to convert them into discrete data, in a discretization preprocessing step. To do this in a privacy-preserving manner, we use a dedicated MPC-based discretization protocol π_{DISC}.

Moreover, the SID3 protocol also requires that its input is in a one-hot-encoding format, which speeds up the computation in dot-product and set intersection. Thus, the secure one-hot-encoding conversion should also be in the preprocessing. Then, what we need to do is mimic the random feature selection and bootstrap sampling in a secure way. We propose the new SMC-based protocol for secure training of a random forest classifier over continuous data that is given in Protocol 3. Except for the existing SID3 subprotocol, there are four major new building blocks: secure discretization (π_{DISC}), secure one-hot-encoding conversion (π_{OHE}), secure random feature selection, and secure bootstrap sampling. This chapter will cover all the details one by one.

4.1 Secure Discretization

Discretization or "binning" is a common form of data preprocessing, aimed at grouping continuous or numerical values into a smaller number of bins. In this thesis, we use *equal-*

Procedure 3: Secure Protocol π_{RF} for Training a Random Forest with Continuous Data.

Input : A secret shared set $[\![S]\!]$ with n training samples (each sample has f features), the number m of trees in an ensemble, the number of buckets p for each feature, the number of features k used in each tree, the number of samples s used in each tree.

Output: A random forest model $[\![RF]\!] = [\![t_1]\!], \ldots, [\![t_m]\!]$.

(Offline Phase) The TI generates and secret shares m matrices $FS^{(1)}, \ldots, FS^{(m)}$ of size $f \cdot p \times k \cdot p$, and m matrices $SS^{(1)}, \ldots, SS^{(m)}$ of size $s \times n$

Discretize each feature of $[\![S]\!]$ into p buckets using protocol π_{DISC} to get $[\![S_{\mathsf{disc}}]\!]$

$[\![S_{\mathsf{OHE}}]\!] \leftarrow \pi_{\mathsf{OHE}}([\![S_{\mathsf{disc}}]\!])$.

for $i \leftarrow 1$ **to** m **do**

    $[\![S_{\mathsf{FS}}]\!] \leftarrow \pi_{\mathsf{DMM}}([\![S_{\mathsf{OHE}}]\!], [\![FS]\!]^{(i)})$.

    $[\![S_{\mathsf{SR}}]\!] \leftarrow \pi_{\mathsf{DMM}}([\![SS]\!]^{(i)}, [\![S_{\mathsf{FS}}]\!])$.

    Use π_{SID3} to securely train a decision tree $[\![t_i]\!]$ with the data set $[\![S_{\mathsf{SR}}]\!]$.

end

return $[\![RF]\!] = [\![t_1]\!], \ldots, [\![t_m]\!]$.

width binning, which means that the range of values for each attribute is divided into a predefined number p of bins of the same width. The computing parties work together to discretize each feature (column) in the training data set S. Initially, all feature values in S are secret shared across the computing parties. After the secure discretization, each party owns secret shares of the discretized data set S_{disc}, and each feature in S_{disc} shares the same number p of categories.

The basic idea behind equal-width binning is that for each feature, we calculate the minimum and maximum of the feature values securely, based on which, we compute the range of each bin and the threshold of each bin. Formally, let D be a vector containing the original attribute values (e.g. the ages of all the users in the data set). The range of D is bounded by the smallest and the largest value occurring in D, i.e. $\min(D)$ and $\max(D)$. To divide this range into p bins of the same width, thresholds need to be placed at

$$h_i = \min(D) + i \cdot \frac{\max(D) - \min(D)}{p} \tag{4.1}$$

Then, doing secure comparison between the threshold of each bin and every value gives each party a vector of binary shares of the comparison result. In order to get the exact value of the bin in the predefined ring [13], we have to convert the binary shares to the ring and do the summation. The overall discretization protocol π_{DISC} is presented in Protocol 4. It contains several subprotocols which we explain next, namely π_{minmax}, π_{geq}, and π_{2toQ}. π_{2toQ} is not covered in this work, but the detail could be found in [13].

4.1.1 Secure MinMax

To compute secret sharings of $\min(D)$ and $\max(D)$ without revealing any information about D, a secure protocol can be formulated similarly to a naive solution in the clear. That is, start by comparing the first and second elements of D to determine an initial estimate of the max and min. Next, iterate through all remaining elements and adjust the max and min estimates when a new largest or smallest element is found. After the n-th element of D is checked, the estimates are guaranteed to be the global min and max. The only necessary adaptation for

Procedure 4: Secure Equal Width Discretization Protocol π_{DISC}

Input : $[\![D]\!]$, public number of buckets p.

Output: $[\![D']\!]$

1 The parties call π_{minmax} on $[\![D]\!]$ and receive $[\![D_{\mathsf{min}}]\!]$, $[\![D_{\mathsf{max}}]\!]$

2 $[\![D_{\mathsf{range}}]\!] \leftarrow [\![D_{\mathsf{max}}]\!] - [\![D_{\mathsf{min}}]\!]$

3 **for** $i \leftarrow 1$ **to** $p-1$ **do**

4 $\quad [\![h_i]\!] \leftarrow [\![D_{\mathsf{min}}]\!] + \pi_{\mathsf{trunc}}(\frac{i}{p} \cdot [\![D_{\mathsf{range}}]\!])$

5 **end**

6 **for** $j \leftarrow 1$ **to** n **do**

7 $\quad$ Let $[\![d'_j]\!] \leftarrow \sum_{i=1}^{p-1} \pi_{\mathsf{2toQ}}(\pi_{\mathsf{geq}}([\![d_j]\!], [\![h_i]\!]))$

8 **end**

9 **return** $[\![D']\!]$

this algorithm to act as an oblivious protocol is to require that the comparisons between the current estimates and each new element of D are performed with π_{geq} (see Section 4.1.2) and that the reassignments are handled with multiplication rather than control flow logic. For example, the comparison based branch operation "**if** $a \geq b$ **then** $b = a$" can be rephrased as

$$\begin{aligned} [\![c]\!] &\leftarrow \pi_{\mathsf{2toQ}}(\pi_{\mathsf{geq}}([\![a]\!], [\![b]\!])) \\ [\![b]\!] &\leftarrow [\![c]\!] \cdot [\![a]\!] + (1 - [\![c]\!]) \cdot [\![b]\!] \end{aligned} \tag{4.2}$$

where c is 1 or 0, depending on the outcome of the comparison $a \geq b$. This form of conditional assignment doesn't allow Alice or Bob to learn anything about which branch of the control flow sequence was followed to arrive at the outcome. An additional detail is that because π_{geq} returns secret shares over $\mathbb{Z}_2$, the result must first be converted to a ring representation with π_{2toQ} before the multiplication can be carried out.

Protocol π_{minmax} has linear communication complexity when carried out in the naive formulation described in Protocol 5. However, it can be improved straightforwardly with the same optimisation technique used for securely computing the repeated product over a

Procedure 5: Secure Min/Max-Finding Protocol π_{minmax}

Input : $[\![D]\!]$, number n of elements in $[\![D]\!]$

Output: $[\![D_{\mathsf{min}}]\!]$, $[\![D_{\mathsf{max}}]\!]$

Let $[\![\geq^?]\!] \leftarrow \pi_{\mathsf{2toQ}}(\pi_{\mathsf{geq}}([\![d_2]\!], [\![d_1]\!]))$

Let $[\![D_{\mathsf{min}}]\!] \leftarrow [\![\geq^?]\!] \cdot [\![d_1]\!] + (1 - [\![\geq^?]\!]) \cdot [\![d_2]\!]$

Let $[\![D_{\mathsf{max}}]\!] \leftarrow [\![\geq^?]\!] \cdot [\![d_2]\!] + (1 - [\![\geq^?]\!]) \cdot [\![d_1]\!]$

for $i \leftarrow 3$ **to** n **do**

    $[\![\geq^?_{\mathsf{min}}]\!] \leftarrow \pi_{\mathsf{2toQ}}(\pi_{\mathsf{geq}}([\![d_i]\!], [\![D_{\mathsf{min}}]\!]))$

    $[\![\geq^?_{\mathsf{max}}]\!] \leftarrow \pi_{\mathsf{2toQ}}(\pi_{\mathsf{geq}}([\![d_i]\!], [\![D_{\mathsf{max}}]\!]))$

    $[\![D_{\mathsf{min}}]\!] \leftarrow [\![\geq^?_{\mathsf{min}}]\!] \cdot [\![D_{\mathsf{min}}]\!] + (1 - [\![\geq^?_{\mathsf{min}}]\!]) \cdot [\![d_i]\!]$

    $[\![D_{\mathsf{max}}]\!] \leftarrow [\![\geq^?_{\mathsf{max}}]\!] \cdot [\![d_i]\!] + (1 - [\![\geq^?_{\mathsf{max}}]\!]) \cdot [\![D_{\mathsf{max}}]\!]$

end

return $[\![D_{\mathsf{min}}]\!]$, $[\![D_{\mathsf{max}}]\!]$

vector of values in which pairwise products are taken until only one value remains [9]. The protocol π_{minmax} is analogous to repeated multiplication because both multiplication and the max/min functions are associative. So, the sequence of many repeated applications can be altered to reduce the total number of consecutive, mutually dependent applications. The basic observation is that the global minimum of D is contained in the set of all pairwise minima of D. Moreover, if the minimum is computed between each pair of entries, and this process is repeated until only one pair remains, the result of the final pairwise comparison is $\min(D)$. The same principle extends to finding the global maximum. See Figure 4.1 for an example.

As a result, π_{minmax} can be computed in a circuit of depth $\lceil \log(n) \rceil$, where at each layer there are $\lceil \log(a+b) \rceil + 1$ rounds of communication.

At the end of this protocol, Alice and Bob have secret sharings of $\min(D)$ and $\max(D)$, which they can then securely combine to compute secret sharings of each of the h_i thresholds

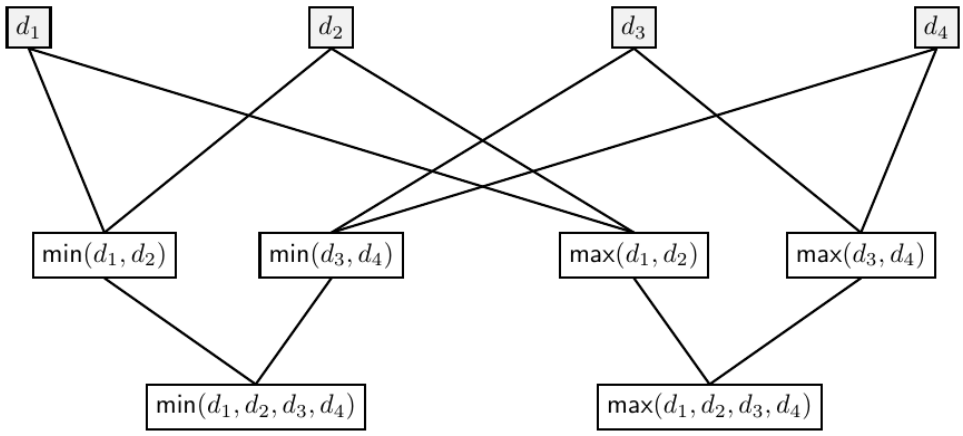

Figure 4.1: Optimised π_{minmax}: An example circuit to compute π_{minmax} on an input vector of size $n = 4$

(see Line 4 in Protocol 4). For example, if $\min(D) = 0$, $\max(D) = 150$, and $p = 6$, then Alice and Bob would compute secret shares of the thresholds $25, 50, 75, 100$, and 125. We assume that p is publicly known, i.e. Alice and Bob know how many bins need to be created. As explained above, they may however not know the value of $\min(D)$ or of $\max(D)$ in the clear, and our discretization protocol does not leak this value.

4.1.2 Secure Comparison

Previous MPC protocols for secure comparison and equality tests between secret shared integers,

$$[\![a]\!] \geq^{?} [\![b]\!] \ : \ [\![1]\!]_2 \ \textbf{else} \ [\![0]\!]_2,$$

$$[\![a]\!] =^{?} [\![b]\!] \ : \ [\![1]\!]_2 \ \textbf{else} \ [\![0]\!]_2,$$

all rely on first converting $\mathbb{Z}_q$-shared integers into bitwise sharings over $\mathbb{Z}_2$. Using state-of-the-art bit decomposition techniques, this step still adds communication rounds logarithmically proportional to the bit length of q; a cost that becomes expensive as the number of consecutive applications of the protocol increases.

Below, we describe protocols for comparison and equality tests over $\mathbb{Z}_{2^\lambda}$ – π_{geq} and π_{eq}, respectively – that circumvent bit decomposition. These protocols are based on a modification of the optimized bit decomposition protocol $\pi_{\mathsf{decompOPT}}$ presented in [13]. These protocols rely on observations that are only true for two's complement representations in $\mathbb{Z}_{2^\lambda}$. This is a relatively weak limitation because even strictly positive data sets can be injected into an integer ring. – e.g. a set of values in the range $[0, 2^{\lambda-1})$ can be injected into the integer ring $\mathbb{Z}_{2^\lambda}$ which represents the range $[-2^{\lambda-1}, 2^{\lambda-1})$. However, it fails for applications that rely on other MPC protocols for which the existence of modular inverses must be guaranteed.

The key insight we use to compute $x \geq^? y$ securely is the following: if $y > x$, then $0 > x - y$. For $x,\ y$ in two's complement form, $0 > x - y \iff \mathsf{MSB}(x - y) = 1$, where $\mathsf{MSB}(\cdot)$ denotes the *most significant bit* of a value. Then,

$$x \geq y \iff \mathsf{MSB}(x - y) = 0.$$

Similar logic can be used to formulate the equality check $x =^? y$:

$$\begin{aligned} x = y &\iff (x \geq y) \wedge (y \geq x) \\ &\iff \mathsf{MSB}(x - y) = \mathsf{MSB}(y - x) = 0 \end{aligned}$$

Note that due to the two's complement form, it is not possible to have $\mathsf{MSB}(x - y) = \mathsf{MSB}(y - x) = 1$. The above shows that the efficiency of computing $x \geq^? y$ and $x =^? y$ is limited only by our ability to extract the most significant bit of a secret-shared value.

Note also that when working with a two's complement fixed-point representation over $\mathbb{Z}_{2^\lambda}$, all bits outside of the injected ring $\mathbb{Z}_{2^{a+b}}$, where a is the number of fractional bits and b is the number of integer bits, are equivalent to the most significant bit. It follows in this case that extracting the $(a + b + 1)$-th bit is sufficient.

The two-party protocol $\pi_{\mathsf{decompOPT}}$ for performing a full decomposition of a $\mathbb{Z}_{2^\lambda}$-shared secret into bitwise sharings over $\mathbb{Z}_2$ is, to our knowledge, the most efficient in the literature. It is based on a *matrix composition network* that computes the difference between each bitwise sum of two secret shares and the corresponding "actual" bit of the secret value in $\mathbb{Z}_{2^\lambda}$. See

[13] for a complete description. An important aspect of this approach is that computing the difference for the α-th bit depends only on $\lceil \log(\alpha - 1) \rceil$ rounds of matrix composition, where each matrix composition requires 4 bits of data transfer. Moreover, the difference for the α-th bit is independent of all results for lower order bits.

In the protocol descriptions that follow, we make use of a straightforward modification of $\pi_{\mathsf{decompOPT}}$ for extracting the α-th bit from a $\mathbb{Z}_{2^\lambda}$-shared secret between two parties, hereafter π_{BTX}. The total number of communication rounds to extract the α-th bit is $\lceil \log(\alpha - 1) \rceil + 1$ with total data transfer of $2(\alpha - 1) + 4\lceil \log(\alpha - 1) \rceil$ bits.

Procedure 6: Secure Protocol π_{BTX} extracts the α-th bit from a secret shared value.

Input : $[\![x]\!]_{2^\lambda},\ \alpha$

Output: $[\![x_\alpha]\!]_2$

Party i regards the lowest α bits of its share, x_i, as $p_{i,1}, \ldots, p_{i,\alpha}$ s.t.

$[\![p_j]\!]_2, = p_{1,j} \oplus p_{2,j}$ for $j = 1, \ldots, \alpha$

Party 1 creates the sharing $[\![g_{1,j}]\!]_2 = (p_{1,j}, 0)$.

Party 2 creates the sharing $[\![g_{2,j}]\!]_2 = (0, p_{2,j})$.

$[\![g_j]\!]_2 \leftarrow [\![g_{1,j}]\!]_2 [\![g_{2,j}]\!]_2$

$$[\![M_j]\!]_2 \leftarrow \begin{bmatrix} [\![p_j]\!]_2 & [\![g_j]\!]_2 \\ 0 & 1 \end{bmatrix} \text{ for } j = 1, ..., \alpha - 1$$

$[\![M_{1.(\alpha-1)}]\!]_2 \leftarrow \mathsf{ComposeNet}_\alpha^{\mathsf{BTX}}([\![M]\!]_2)$

$[\![c_{\alpha-1}]\!]_2 \leftarrow$ the upper right entry of $[\![M_{1.(\alpha-1)}]\!]_2$

return $[\![p_\alpha]\!]_2 \oplus [\![c_{\alpha-1}]\!]_2$

4.2 Secure One-hot-encoding Conversion

For securely training each decision tree over the discretized data we will use the secure ID3 protocol π_{SID3} from de Hoogh et al. [14]. That protocol takes the data in a one-hot-encoding (OHE) format, so we need to convert to OHE as an intermediate step. The $n \times f$ data

Procedure 7: Secure Protocol π_{geq} computes the integer comparison of two secrets in $\mathbb{Z}_{2^\lambda}$.

Input : $[\![x]\!]$, $[\![y]\!]$, $\alpha :=$ the lowest bit position equivalent to the MSB.

Output: $[\![x]\!] \geq^? [\![y]\!]$: $[\![1]\!]_2$ **else** $[\![0]\!]_2$

1 Let $[\![\mathsf{diff}]\!] \leftarrow [\![x]\!] - [\![y]\!]$

2 Let $[\![\mathsf{MSB}]\!]_2 \leftarrow \pi_{\mathsf{BTX}}([\![\mathsf{diff}]\!], \alpha)$

3 **return** $1 \oplus [\![MSB]\!]_2$

Procedure 8: Secure Protocol π_{eq} computes the integer equality test on two secrets in $\mathbb{Z}_{2^\lambda}$.

Input : $[\![x]\!]$, $[\![y]\!]$, $\alpha :=$ the lowest bit position equivalent to the MSB.

Output: $[\![x]\!] =^? [\![y]\!]$: $[\![1]\!]_2$ **else** $[\![0]\!]_2$

1 Let $[\![\mathsf{d}_1]\!] \leftarrow [\![x]\!] - [\![y]\!]$ and $[\![\mathsf{d}_2]\!] \leftarrow [\![y]\!] - [\![x]\!]$

2 Let $[\![\mathsf{MSB}_1]\!]_2 \leftarrow \pi_{\mathsf{BTX}}([\![\mathsf{d}_1]\!], \alpha)$ and $[\![\mathsf{MSB}_2]\!]_2 \leftarrow \pi_{\mathsf{BTX}}([\![\mathsf{d}_2]\!], \alpha)$// run in parallel

3 **return** $1 \oplus [\![MSB_1]\!]_2 \oplus [\![MSB_2]\!]_2$

matrix S_{disc} is converted to an $n \times (f \cdot p)$ matrix S_{OHE}. In both matrices, each row represents an instance. In S_{disc}, each column represents a feature, while in S_{OHE}, p columns are used to represent that same feature. For example, say that we have a data set with $n = 4$ instances and $f = 3$ features, and that after discretization into $p = 3$ bins per feature, the parties have secret shares of the values in S_{disc}. As said above, in S_{disc}, each row corresponds to an instance, and each column to a feature.

$$S_{\mathsf{disc}} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 2 \end{pmatrix} \qquad S_{\mathsf{OHE}} = \begin{pmatrix} 0\,1\,0 & 1\,0\,0 & 1\,0\,0 \\ 0\,0\,1 & 0\,0\,1 & 0\,1\,0 \\ 1\,0\,0 & 0\,1\,0 & 0\,0\,1 \\ 1\,0\,0 & 0\,1\,0 & 0\,0\,1 \end{pmatrix} \tag{4.3}$$

After protocol π_{OHE}, the parties need to hold secret shares of the corresponding OHE representation S_{OHE}. To convert S_{disc} into S_{OHE} in an oblivious manner, the parties first

duplicate each column of S_{disc} p times, to create secret shares of $S^{\text{dup}}_{\mathsf{disc}}$. Next, each of the $n \times f \cdot p$ values of $S^{\text{dup}}_{\mathsf{disc}}$ is compared, using the secure equality test π_{eq}, to the corresponding entry in the mask matrix M, and the result of this test, which are secret shares of 0 or 1, is recorded in the corresponding position in S_{OHE}.

The protocol π_{OHE} for oblivious conversion from categorical values to OHE is presented in Protocol 9. The equality checks in Line 9 can be done in a batch for efficiency.

$$S^{\text{dup}}_{\mathsf{disc}} = \begin{pmatrix} 1\,1\,1 & 0\,0\,0 & 0\,0\,0 \\ 2\,2\,2 & 2\,2\,2 & 1\,1\,1 \\ 0\,0\,0 & 1\,1\,1 & 2\,2\,2 \\ 0\,0\,0 & 1\,1\,1 & 2\,2\,2 \end{pmatrix} \quad M = \begin{pmatrix} 0\,1\,2 & 0\,1\,2 & 0\,1\,2 \\ 0\,1\,2 & 0\,1\,2 & 0\,1\,2 \\ 0\,1\,2 & 0\,1\,2 & 0\,1\,2 \\ 0\,1\,2 & 0\,1\,2 & 0\,1\,2 \end{pmatrix} \tag{4.4}$$

Procedure 9: Secure Conversion to One-Hot-Encoding with Protocol π_{OHE}.

Input : $[\![T]\!]$.

Output: $[\![T_{\mathsf{OHE}}]\!]$

1 Let $[\![T^{\text{dup}}]\!] \leftarrow$ an empty matrix
2 Let $[\![M]\!] \leftarrow$ an empty matrix
3 **for** $[\![col]\!] \leftarrow [\![columns]\!]$ ***in*** $[\![T]\!]$ **do**
4 **for** $i \leftarrow 0$ **to** $p - 1$ **do**
5 Append $[\![col]\!]$ to $[\![T^{\text{dup}}]\!]$
6 Append $[\![[i, i, ..i]^{\text{T}}]\!]$ to $[\![M]\!]$
7 **end**
8 **end**
9 Let $[\![T_{\mathsf{OHE}}]\!] \leftarrow \pi_{\mathsf{eq}}([\![T^{\text{dup}}]\!], [\![M]\!])$
10 **return** $[\![T_{\mathsf{OHE}}]\!]$

4.3 Secure Feature Selection and Secure Bootstrap Sampling

After converting to OHE, and before invoking π_{SID3} to train each tree in the random forest, we need to randomly select the features to be used in that tree and to perform the random sampling with replacement of the instances.

In order to randomly select the features we will use a selection matrix that is generated by the TI and secret shared with the parties. This selection matrix is multiplied with the data set to extract the desired features. For a standard matrix-like format data set, i.e., $instances \times features$, the feature selection matrix FS would simply be a matrix in which each column contains a one in the position corresponding to the index of the selected feature and zeroes in the other positions. Note that each row also has only one non-zero entry as the selected features are different. For example, for the data set S_{disc} described above, to select the first and third features, the selection matrix would be

$$FS = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

and the result

$$S_{\mathsf{FS}} = S_{\mathsf{disc}} \times FS = \begin{pmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 2 \\ 0 & 2 \end{pmatrix}.$$

For oblivious selection of k features from S_{OHE}, we use a $f \cdot p \times k \cdot p$ selection matrix FS that is generated by the TI and secret shared with the parties. This selection matrix is multiplied with the data set to extract the desired features. For example, multiplying S_{OHE}

from Equation (4.3) with

$$FS = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.5}$$

will retain the first 3 and the last 3 columns of S_{OHE}, thereby effectively selecting the first and third feature of S_{disc}.

Note that in FS, identity matrices of size $p \times p$ are used to indicate the selections, and the remaining positions are filled with zeroes. m such matrices are randomly populated in this manner by the TI, with m the total number of DT in the RF. The secure protocol for random feature selection π_{RFS} simply multiplies the OHE-style data set S_{OHE} with a feature selection matrix FS that is secret shared by the TI using the secure matrix multiplication protocol π_{DMM}.

The procedure used to sample the s instances with replacement is similarly done using a multiplication with a $s \times n$ matrix SS that is secret shared by the TI. The only difference is that the identity matrices used for selecting can be aligned (as the choice is with replacement). We denote this result by S_{SR}.

Chapter 5

IMPLEMENTATION AND EXPERIMENTS

We have implemented the protocols proposed in this thesis in the Rust[1] programming language. It is a powerful lower-level programming langauge that is popular in the cryptography community because of its efficiency and reliability in thread security. Furthermore, this work would be merged into our open source PPML framework.

Table 5.1 contains accuracy and runtime results for three different data sets. All data sets are for binary classification problems. The number of continuous valued input features, as well as the number of instances in each data set is mentioned in the table, along with the parameter λ that determines the size of the ring $\mathbb{Z}_{2^\lambda}$. The parameters a and b denote the number of bits used to represent the fractional and the integer part of continuous feature values when converted to a value in $\mathbb{Z}_{2^\lambda}$, as explained in Chapter 3.

All runtime results in this chapter (in minutes) were obtained on a Macbook Pro machine with 8 core i7 CPUs, 16.0 GiB Memory. We use different processes with different ports to simulate separate parties working together. The communication time would increase when deployed in the real world, but the computing time would decrease as the capability of computers engaged improve. So here we are going to analyze the accuracy change and running time increase in different set of hyperparameters for different datasets.

As is shown in the table 5.1, the accuracy in such hyperparameter setting could be at least very close to the accuracy achieved by Decision Tree Training in the clear. In these

---

[1]https://www.rust-lang.org/

[1]https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

[2]https://www.kaggle.com/shayanfazeli/heartbeat

[3]https://www.kaggle.com/sammy123/lower-back-pain-symptoms-dataset

data set	# instances	# features	DT Accuracy	Random Forest	
				Acc	Time
Lower Back Pain Symptoms [3]	310	12	0.63	0.69	0.17
Breast Cancer [1]	569	30	0.92	0.92	0.3
ECG Heartbeat [2]	14,552	187	0.82	0.72	50

Table 5.1: Overview of data sets, accuracy (5-fold cross-validation), and runtime results (in min) for secure training of decision tree based models. The details of hyperparameters are as follows: #bins $p = 4$, #trees $m = 100$, tree depth $d = 1$, $k = \sqrt{\#features}$,$s = \sqrt{\#instances}$

experiments, $a = 10$,$b = 15$,$\lambda = 64$. The running time of each experiment depends on the size of the dataset. Even for the largest dataset, the training took less than an hour.

The accuracy and runtime results of course depend on several hyperparameter choices, such as the number of bins used for discretization, the depth of the decision trees, and the number of trees in the ensembles. Below, we provide a more detailed analysis of the effect of each of these hyperparameters on the accuracy and runtime over the Breast Cancer data set.

For each experiment shown in Table 5.5-5.7 below, four hyperparameters are varied to produce a new combination, which are:

- m, the number of trees
- p, the number of bins
- k, the number of features selected per tree
- s, the number of instances selected per tree

All the accuracy and runtime numbers are average results obtained from 5-fold cross-validation.

We include three sets of experiments in this section. For each set, we fixed other hyperparameters except two. One of the two hyperparameters is our main variable that impacts the accuracy and runtime, while the other one is to help whether that factor could impact

the tendency of the change, e.g., whether that parameter would change the shape of the line in figures.

m	p	k	s	Time (min)	Acc
100	3	5	21	0.24	0.9121
200	3	5	21	0.46	0.9209
300	3	5	21	0.63	0.9086
400	3	5	21	0.80	0.9104
500	3	5	21	0.88	0.9086

Table 5.2: Accuracy and runtime results of π_{RF} protocol for secure training of a random forest classifier on the Breast Cancer data set, where m is a variable, $p = 3$

m	p	k	s	Time (min)	Acc
100	4	5	21	0.30	0.9173
200	4	5	21	0.52	0.9173
300	4	5	21	0.75	0.9155
400	4	5	21	0.95	0.9155
500	4	5	21	1.21	0.9155

Table 5.3: Accuracy and runtime results of π_{RF} protocol for secure training of a random forest classifier on the Breast Cancer data set, where m is a variable, $p = 4$

From figures 5.1-5.2, we can see that the running time increases linearly with the number of trees no matter how many bins are used for discretization. Furthermore, the increase rate stays almost the same for all kinds of bins. The overall running time stays low even for a high number of bins. Also, from Figure 5.1, we can infer that, for this data set, the number of trees doesn't affect the accuracy a lot when $m \geq 100$.

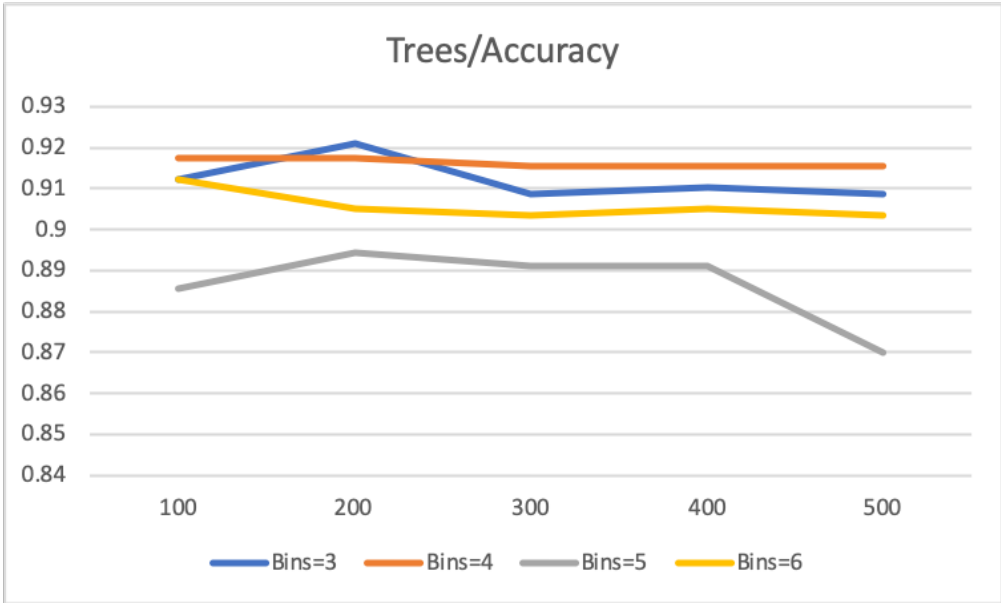

Figure 5.1: Number of trees for different bins versus Accuracy

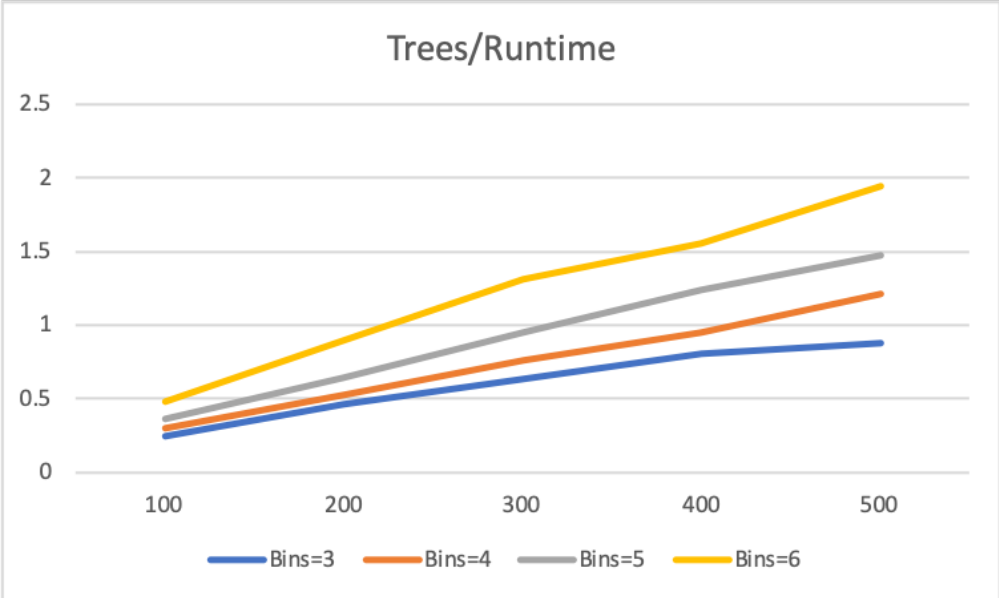

Figure 5.2: Number of trees for different bins versus Runtime

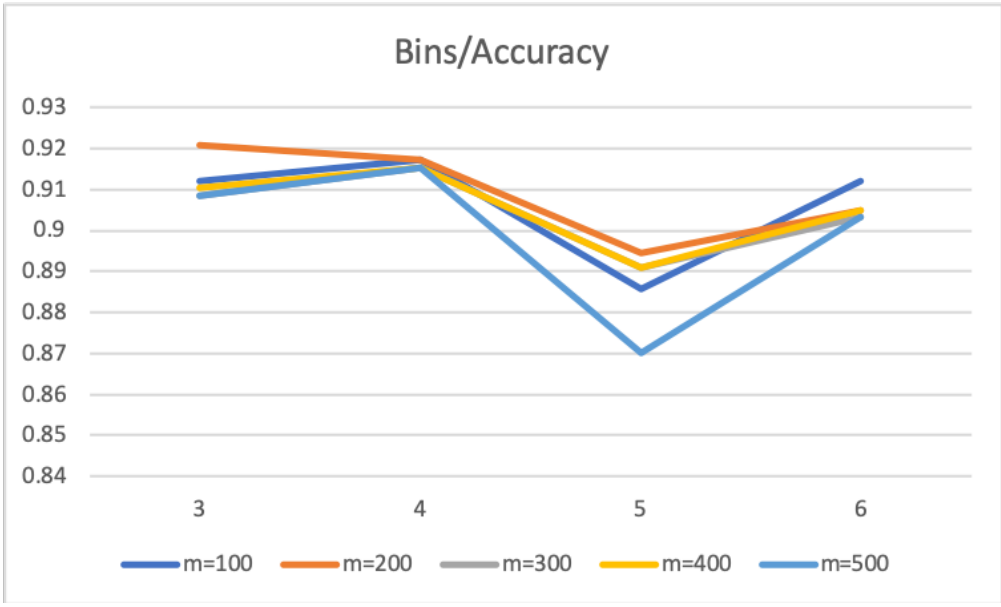

Figure 5.3: Number of bins for different number of trees versus Accuracy

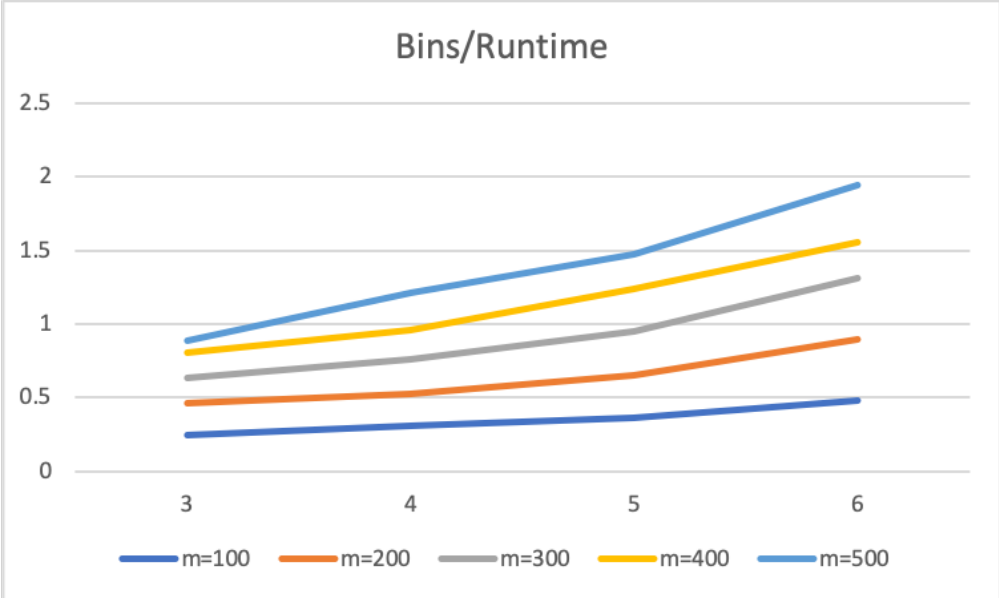

Figure 5.4: Number of bins for different number of trees versus Runtime

m	p	k	s	Time (min)	Acc
100	5	5	21	0.36	0.8857
200	5	5	21	0.64	0.8945
300	5	5	21	0.95	0.891
400	5	5	21	1.23	0.891
500	5	5	21	1.47	0.87

Table 5.4: Accuracy and runtime results of π_{RF} protocol for secure training of a random forest classifier on the Breast Cancer data set, where m is a variable, $p = 5$

Figure 5.3 and 5.4 display how the number of bins affect the accuracy and runtime. We have four different numbers of bins, i.e., $p = 3$, $p = 4$, $p = 5$, and $p = 6$. We observe a drop in accuracy when $p = 5$. This may be specific to this dataset, but it still worth further investigation.

Next we experimentally evaluate the effect of different numbers of features selected on the accuracy and running time. As can be seen in Table 5.6 and Figure 5.5 and 5.6, the value of k does have an impact on both accuracy and running time.The accuracy first increases from $k = 5$ to $k = 15$, then it drops from $k = 15$ to $k = 30$. When the number of features increases, the bias decreases and the variance increases, which causes overfitting when $k = 30$. The running time seems to increase at a different rate. The number of features is a quite crucial factor that affects both running time and accuracy, which we could consider as a major direction when improving and training.

A similar phenomenon occurs when varying the number of samples, as shown in Table 5.7 and Figure 5.7 and 5.8. The RFs overfit the training data when we increase the value of s to 455, which is the number of elements in the training portion of the data set.

m	p	k	s	Time (min)	Acc
100	6	5	21	0.48	0.9121
200	6	5	21	0.89	0.905
300	6	5	21	1.31	0.9033
400	6	5	21	1.55	0.905
500	6	5	21	1.94	0.9033

Table 5.5: Accuracy and runtime results of π_{RF} protocol for secure training of a random forest classifier on the Breast Cancer data set, where m is a variable, $p = 6$

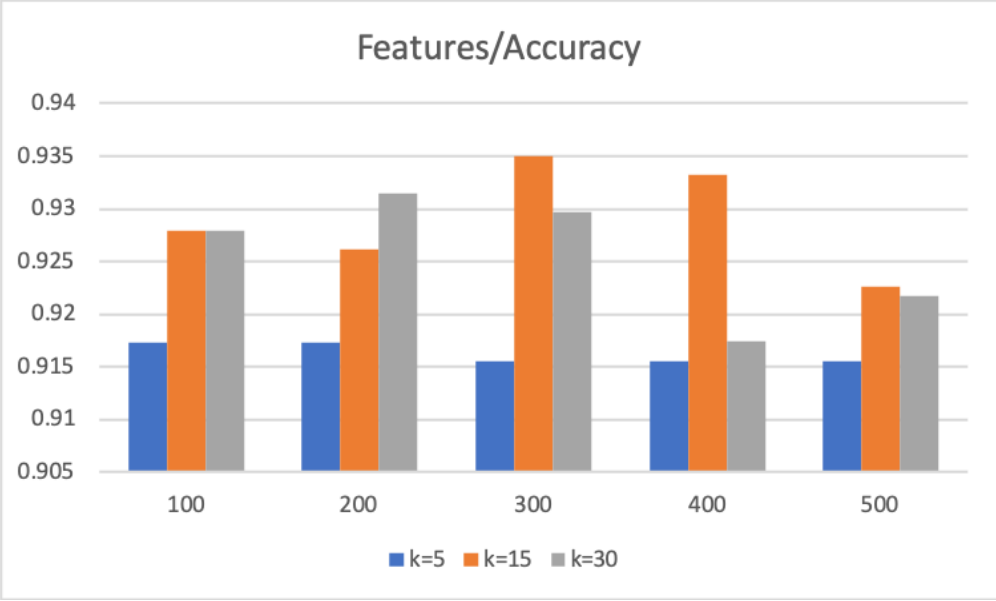

Figure 5.5: Number of features selected for different bins versus Accuracy

m	p	k	s	Time (min)	Acc
100	4	5	21	0.30	0.9173
100	4	15	21	0.85	0.9279
100	4	30	21	2.21	0.9279
200	4	5	21	0.52	0.9173
200	4	15	21	1.74	0.9261
200	4	30	21	4.53	0.9314
300	4	5	21	0.75	0.9155
300	4	15	21	2.78	0.935
300	4	30	21	5.84	0.9297
400	4	5	21	0.95	0.9155
400	4	15	21	3.71	0.9332
400	4	30	21	9.11	0.9174
500	4	5	21	1.21	0.9155
500	4	15	21	4.90	0.9226
500	4	30	21	10.98	0.9217

Table 5.6: Accuracy and runtime results of π_{RF} protocol for secure training of a random forest classifier on the Breast Cancer data set, where k is a variable when m is also fixed

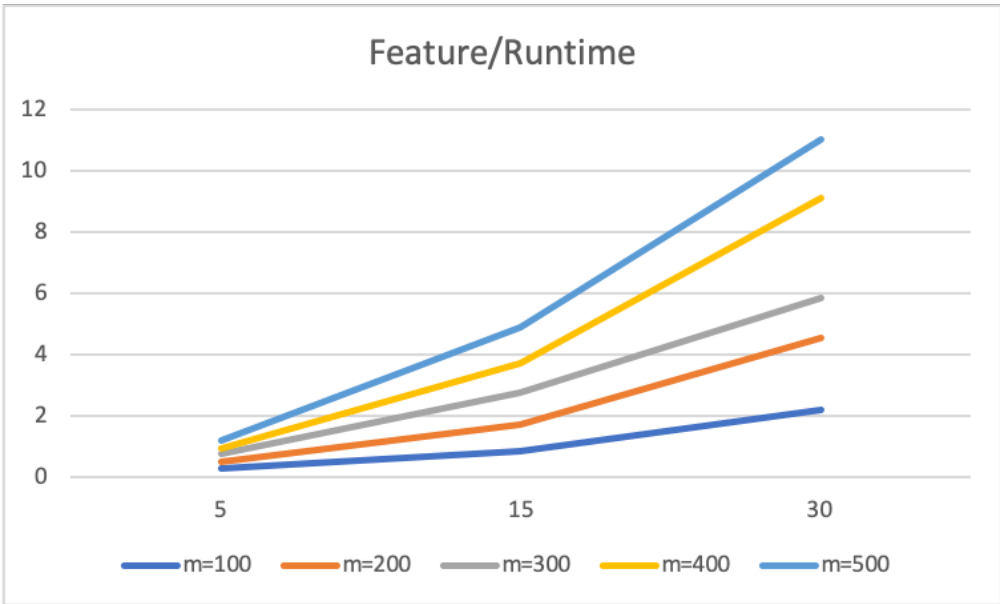

Figure 5.6: Number of features selected for different bins versus Runtime

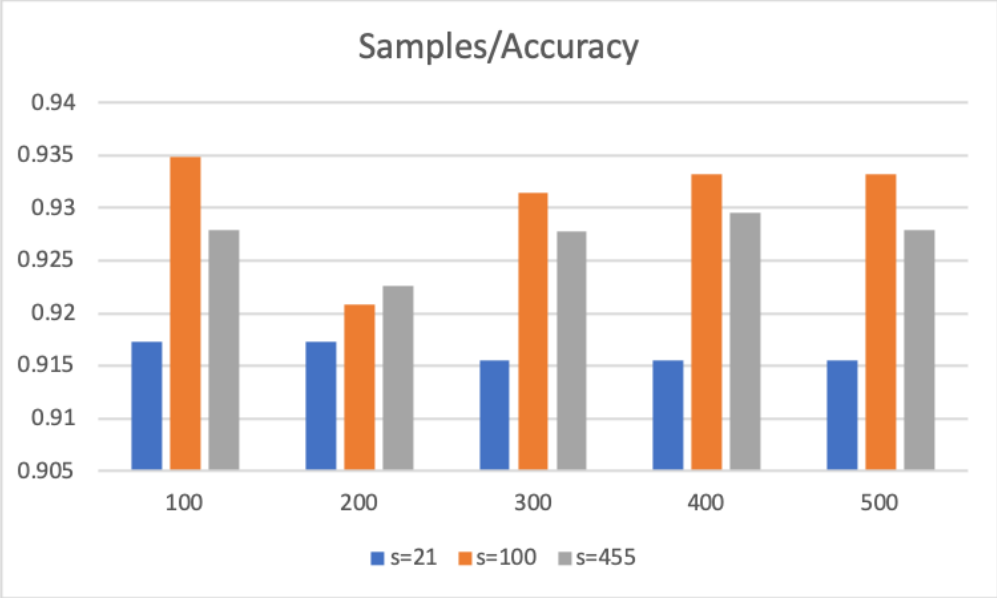

Figure 5.7: Number of samples selected for different bins versus Accuracy

m	p	k	s	Time (min)	Acc
100	4	5	21	0.30	0.9173
100	4	5	100	0.45	0.9349
100	4	5	455	1.02	0.9279
200	4	5	21	0.52	0.9173
200	4	5	100	0.97	0.9208
200	4	5	455	2.17	0.9226
300	4	5	21	0.75	0.9155
300	4	5	100	1.22	0.9314
300	4	5	455	3.732	0.9278
400	4	5	21	0.95	0.9155
400	4	5	100	1.67	0.9332
400	4	5	455	4.40	0.9296
500	4	5	21	1.21	0.9155
500	4	5	100	2.16	0.9332
500	4	5	455	5.91	0.9279

Table 5.7: Accuracy and runtime results of π_{RF} protocol for secure training of a random forest classifier on the Breast Cancer data set, where s is a variable when m is also fixed

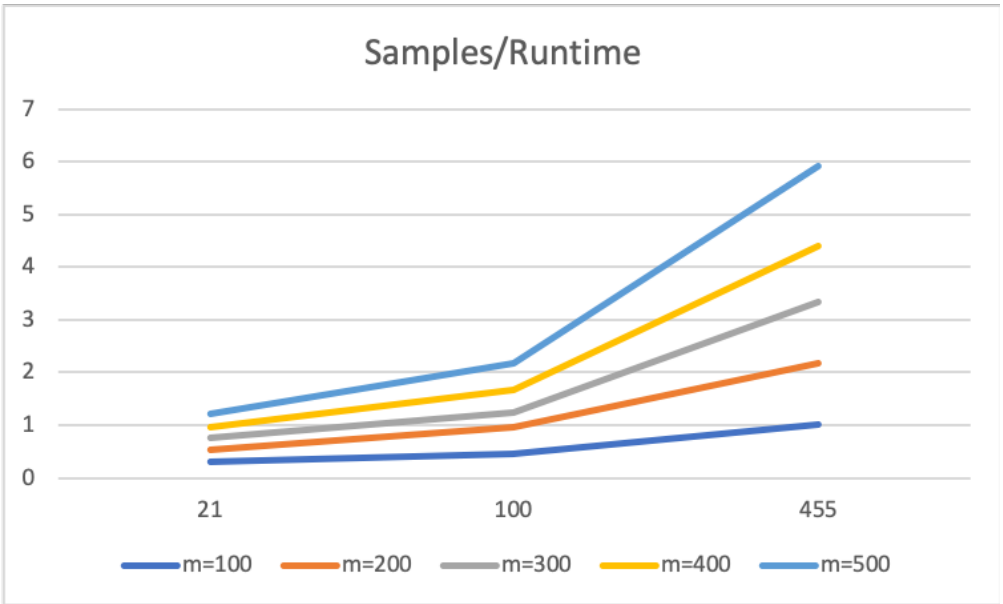

Figure 5.8: Number of samples selected for different bins versus Runtime

Chapter 6

CONCLUSIONS

In this thesis, we have introduced the very first MPC-based protocol for training a random forest classifier in a privacy-preserving manner over data that is distributed across different data owners. Furthermore, our approach can securely train such random forest classifiers over data with continuous feature values. The process is split into two phases: a preprocessing phase, and the training of the classifier.

In the preprocessing stage, we first discretize the continuous data into discrete data with the same number of bins or levels per feature. To achieve that, we use a novel secure discretization protocol that in itself relies on a new secure comparison protocol, and a new secure minimum-maximum protocol. Because of the input format requirement of SID3, which is the underlying secure version of ID3 for training a decision tree, we also propose a secure protocol for conversion of discrete data to one-hot-encoding.

In the training process, we mimic random feature selection and bootstrap sampling for each tree by secure matrix multiplication of the data matrix with feature selection matrices and sampling matrices. These matrices are randomly generated by a trusted initializer (TI) and secret shared among the computing parties. Then we apply the SID3 potocol to train decision trees in parallel.

To validate our work, we implemented the proposed protocols in Rust-Lynx, an open source framework for privacy-preserving machine learning, and we applied them to securely train random forests on several datasets. With my implementation, you can securely train a random forest in less than an hour even on a dataset with approximately 15,000 training examples and 200 features. Besides, compared with the decision tree classifier training using C4.5 algorithm, this work could achieve the same accuracy. For instance, the accuracy

against the Breast Cancer dataset is the same as that of training a decision tree in the clear.

There are several interesting directions for further research. First of all, we could bring the random feature selection and bootstrapping to the preprocessing phase and put the feature selection process in front of secure one-hot-encoding. Second, since our work is based on the semi-honest-party setting, we could extend the work to the settings where parties could be malicious. Third, secure regression tree training and random forest regressors are also two interesting directions.

BIBLIOGRAPHY

[1] Javed A Aslam, Raluca A Popa, and Ronald L Rivest. On estimating the size and confidence of a statistical audit. *EVT*, 7:8, 2007.

[2] Donald Beaver. Commodity-based cryptography. In *STOC*, volume 97, pages 446–455, 1997.

[3] Gopal Behera. Privacy preserving C4.5 using Gini index. In *2nd National Conference on Emerging Trends and Applications in Computer Science*, pages 1–4, 2011.

[4] George Robert Blakley and Catherine Meadows. Security of ramp schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 242–268. Springer, 1984.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Bernardo David, Rafael Dowsley, Raj Katti, and Anderson CA Nascimento. Efficient unconditionally secure comparison and privacy preserving machine learning classification protocols. In *International Conference on Provable Security*, pages 354–367. Springer, 2015.

[7] Bernardo David, Rafael Dowsley, Jeroen van de Graaf, Davidson Marques, Anderson C. A. Nascimento, and Adriana C. B. Pinto. Unconditionally secure, universally composable privacy preserving linear algebra. *IEEE Transactions on Information Forensics and Security*, 11(1):59–73, 2016.

[8] M. De Cock, R. Dowsley, A. Nascimento, D. Railsback, J. Shen, and A. Todoki. Fast secure logistic regression for high dimensional gene data. In *Privacy in Machine Learning workshop (PriML2019) - NeurIPS Workshop*, 2019.

[9] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson Nascimento, Wing-Sea Poon, and Stacey 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, 2019.

[10] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson CA Nascimento, Wing-Sea Poon, and Stacey 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] Martine De Cock, Rafael Dowsley, Anderson C. A. Nascimento, and Stacey C. Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *8th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 3–14, 2015.

[12] Martine De Cock, Rafael Dowsley, Anderson CA Nascimento, and Stacey C Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2015.

[13] Martine De Cock, Rafael Dowsley, Anderson CA Nascimento, Davis Railsback, Jianwei Shen, and Ariel Todoki. High performance logistic regression for privacy-preserving genome analysis. *arXiv preprint arXiv:2002.05377*, 2020.

[14] Sebastiaan de Hoogh, Berry Schoenmakers, Ping Chen, and Harm 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.

[15] Thomas G Dietterich. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems*, volume 1857 of *LNCS*, pages 1–15. Springer, 2000.

[16] Rafael Dowsley. *Cryptography Based on Correlated Data: Foundations and Practice*. PhD thesis, Karlsruhe Institute of Technology, Germany, 2016.

[17] Rafael Dowsley, Jörn Müller-Quade, and Tobias Nilges. Weakening the isolation assumption of tamper-proof hardware tokens. In Anja Lehmann and Stefan Wolf, editors, *ICITS 15: 8th International Conference on Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 197–213, Lugano, Switzerland, May 2–5, 2015. Springer, Heidelberg, Germany.

[18] Rafael Dowsley, Jörn Müller-Quade, Akira Otsuka, Goichiro Hanaoka, Hideki Imai, and Anderson C. A. Nascimento. Universally composable and statistically secure verifiable secret sharing scheme based on pre-distributed data. *IEICE Transactions*, 94-A(2):725–734, 2011.

[19] David Evans, Vladimir Kolesnikov, Mike Rosulek, et al. A pragmatic introduction to secure multi-party computation. *Foundations and Trends in Privacy and Security*, 2(2-3):70–246, 2018.

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[21] Kyle Fritchman, Keerthanaa Saminathan, Rafael Dowsley, Tyler Hughes, Martine De Cock, Anderson Nascimento, and Ankur Teredesai. Privacy-preserving scoring of tree ensembles: A novel framework for AI in healthcare. In *IEEE Big Data*, pages 2413–2422, 2018.

[22] Chuan Guo, Awni Hannun, Brian Knott, Laurens van der Maaten, Mark Tygert, and Ruiyu Zhu. Secure multiparty computations in floating-point arithmetic. *arXiv preprint arXiv:2001.03192*, 2020.

[23] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pages 36–54, 2000.

[24] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[25] Payman Mohassel and Yupeng Zhang. SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 19–38, 2017.

[26] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[27] J Ross Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014.

[28] RC Quinlan. 4.5: Programs for machine learning morgan kaufmann publishers inc. *San Francisco, USA*, 1993.

[29] Devin Reich, Ariel Todoki, Rafael Dowsley, Martine De Cock, and Anderson Nascimento. Privacy-preserving classification of personal text messages with secure multiparty computation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3752–3764, 2019.

[30] Saeed Samet and Ali Miri. Privacy preserving ID3 using Gini index over horizontally partitioned data. In *2008 IEEE/ACS International Conference on Computer Systems and Applications*, pages 645–651, 2008.

[31] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[32] Yanguang Shen, Hui Shao, and Li Yang. Privacy preserving C4.5 algorithm over vertically distributed datasets. In *International Conference on Networks Security, Wireless Communications and Trusted Computing*, volume 2, pages 446–448, 2009.

[33] Rafael Tonicelli, Anderson C. A. Nascimento, Rafael Dowsley, Jörn Müller-Quade, Hideki Imai, Goichiro Hanaoka, and Akira Otsuka. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *International Journal of Information Security*, 14(1):73–84, 2015.

[34] Jaideep Vaidya and Chris Clifton. Privacy-preserving decision trees over vertically partitioned data. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 139–152, 2005.

[35] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 1:24, 2019.

[36] Ming-Jun Xiao, Kai Han, Liu-Sheng Huang, and Jing-Yuan Li. Privacy preserving C4.5 algorithm over horizontally partitioned data. In *2006 Fifth International Conference on Grid and Cooperative Computing (GCC'06)*, pages 78–85, 2006.